CHAPTER

FOUR

PAIR STYLES

4.1 pair_style adp command

Accelerator Variants: *adp/kk*, *adp/omp*

4.1.1 Syntax

```
pair_style adp
```

4.1.2 Examples

```
pair_style adp
pair_coeff * * Ta.adp Ta
pair_coeff * * ../potentials/AlCu.adp Al Al Cu
```

4.1.3 Description

Style *adp* computes pairwise interactions for metals and metal alloys using the angular dependent potential (ADP) of *(Mishin)*, which is a generalization of the *embedded atom method (EAM) potential*. The LAMMPS implementation is discussed in *(Singh)*. The total energy Ei of an atom I is given by

$$E_i = F_\alpha \left(\sum_{j \neq i} \rho_\beta(r_{ij}) \right) + \frac{1}{2} \sum_{j \neq i} \phi_{\alpha\beta}(r_{ij}) + \frac{1}{2} \sum_s (\mu_i^s)^2 + \frac{1}{2} \sum_{s,t} (\lambda_i^{st})^2 - \frac{1}{6} \nu_i^2$$

$$\mu_i^s = \sum_{j \neq i} u_{\alpha\beta}(r_{ij}) r_{ij}^s$$

$$\lambda_i^{st} = \sum_{j \neq i} w_{\alpha\beta}(r_{ij}) r_{ij}^s r_{ij}^t$$

$$\nu_i = \sum_s \lambda_i^{ss}$$

where F is the embedding energy which is a function of the atomic electron density ρ, ϕ is a pair potential interaction, α and β are the element types of atoms I and J, and s and $t = 1, 2, 3$ and refer to the cartesian coordinates. The μ and λ terms represent the dipole and quadruple distortions of the local atomic environment which extend the original EAM framework by introducing angular forces.

Note that unlike for other potentials, cutoffs for ADP potentials are not set in the pair_style or pair_coeff command; they are specified in the ADP potential files themselves. Likewise, the ADP potential files list atomic masses; thus you do not need to use the *mass* command to specify them.

ADP potentials are available from:

- The NIST WWW site at https://www.ctcms.nist.gov/potentials. Note that ADP potentials obtained from NIST must be converted into the extended DYNAMO *setfl* format discussed below.
- The OpenKIM Project at https://openkim.org/browse/models/by-type provides ADP potentials that can be used directly in LAMMPS with the *kim command* interface.

Only a single pair_coeff command is used with the *adp* style which specifies an extended DYNAMO *setfl* file, which contains information for *M* elements. These are mapped to LAMMPS atom types by specifying *N* additional arguments after the filename in the pair_coeff command, where *N* is the number of LAMMPS atom types:

- filename
- *N* element names = mapping of extended *setfl* elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, the potentials/AlCu.adp file, included in the potentials directory of the LAMMPS distribution, is an extended *setfl* file which has tabulated ADP values for w elements and their alloy interactions: Cu and Al. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Al, and the fourth to be Cu, you would use the following pair_coeff command:

```
pair_coeff * * AlCu.adp Al Al Al Cu
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Al arguments map LAMMPS atom types 1,2,3 to the Al element in the extended *setfl* file. The final Cu argument maps LAMMPS atom type 4 to the Al element in the extended *setfl* file. Note that there is no requirement that your simulation use all the elements specified by the extended *setfl* file.

If a mapping value is specified as NULL, the mapping is not performed. This can be used when an *adp* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Adp files in the *potentials* directory of the LAMMPS distribution have an ".adp" suffix. A DYNAMO *setfl* file extended for ADP is formatted as follows. Basically it is the standard *setfl* format with additional tabulated functions u and w added to the file after the tabulated pair potentials. See the *pair_eam* command for further details on the *setfl* format.

- lines 1,2,3 = comments (ignored)
- line 4: N_{elements} Element1 Element2 ... ElementN
- line 5: N_ρ, d_ρ, N_r, d_r, cutoff

Following the 5 header lines are N_{elements} sections, one for each element, each with the following format:

- line 1 = atomic number, mass, lattice constant, lattice type (e.g. FCC)
- embedding function $F(\rho)$ (N_ρ values)
- density function $\rho(r)$ (N_r values)

Following the N_{elements} sections, N_r values for each pair potential $\phi(r)$ array are listed for all i, j element pairs in the same format as other arrays. Since these interactions are symmetric ($i, j = j, i$) only ϕ arrays with $i \geq j$ are listed, in the following order:

$$i, j = (1,1), (2,1), (2,2), (3,1), (3,2), (3,3), (4,1), ..., (N_{\text{elements}}, N_{\text{elements}}).$$

The tabulated values for each ϕ function are listed as $r * \phi$ (in units of eV-Angstroms), since they are for atom pairs, the same as for *other EAM files*.

After the $\phi(r)$ arrays, each of the $u(r)$ arrays are listed in the same order with the same assumptions of symmetry. Directly following the $u(r)$, the $w(r)$ arrays are listed. Note that $\phi(r)$ is the only array tabulated with a scaling by r.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.1.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, no special mixing rules are needed, since the ADP potential files specify alloy interactions explicitly.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in tabulated potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.1.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package.

4.1.6 Related commands

pair_coeff, *pair_eam*

4.1.7 Default

none

(Mishin) Mishin, Mehl, and Papaconstantopoulos, Acta Mater, 53, 4029 (2005).

(Singh) Singh and Warner, Acta Mater, 58, 5797-5805 (2010),

4.2 pair_style agni command

Accelerator Variants: *agni/omp*

4.2.1 Syntax

```
pair_style agni
```

4.2.2 Examples

```
pair_style      agni
pair_coeff      * * Al.agni Al
```

4.2.3 Description

Style *agni* style computes the many-body vectorial force components for an atom as

$$F_i^u = \sum_t^{N_t} \alpha_t \cdot \exp\left[-\frac{\left(d_{i,t}^u\right)^2}{2l^2}\right]$$

$$d_{i,t}^u = ||V_i^u(\eta) - V_t^u(\eta)||$$

$$V_i^u(\eta) = \sum_{j \neq i} \frac{r_{ij}^u}{r_{ij}} \cdot e^{-\left(\frac{r_{ij}}{\eta}\right)^2} \cdot f_d\left(r_{ij}\right)$$

$$f_d\left(r_{ij}\right) = \frac{1}{2}\left[\cos\left(\frac{\pi r_{ij}}{R_c}\right) + 1\right]$$

u labels the individual components, i.e. x, y or z, and V is the corresponding atomic fingerprint. d is the Euclidean distance between any two atomic fingerprints. A total of N_t reference atomic environments are considered to construct the force field file. α_t and l are the weight coefficients and length scale parameter of the non-linear regression model.

The method implements the recently proposed machine learning access to atomic forces as discussed extensively in the following publications - *(Botu1)* and *(Botu2)*. The premise of the method is to map the atomic environment numerically into a fingerprint, and use machine learning methods to create a mapping to the vectorial atomic forces.

Only a single pair_coeff command is used with the *agni* style which specifies an AGNI potential file containing the parameters of the force field for the needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of AGNI elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the force field file.

An AGNI force field is fully specified by the filename which contains the parameters of the force field, i.e., the reference training environments used to construct the machine learning force field. Example force field and input files are provided in the examples/PACKAGES/agni directory.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.2.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.2.5 Restrictions

Currently, only elemental systems are implemented. Also, the method only provides access to the forces and not energies or stresses. The lack of potential energy data makes this pair style incompatible with several of the *minimizer algorthms* like *cg* or *sd*. It should work with damped dynamics based minimizers like *fire* or *quickmin*. However, one can access the energy via thermodynamic integration of the forces as discussed in *(Botu3)*. This pair style is part of the MISC package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The AGNI force field files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the AGNI potential with any LAMMPS units, but you would need to create your own AGNI potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.2.6 Related commands

pair_coeff

4.2.7 Default

none

(Botu1) V. Botu and R. Ramprasad, Int. J. Quant. Chem., 115(16), 1074 (2015).

(Botu2) V. Botu and R. Ramprasad, Phys. Rev. B, 92(9), 094306 (2015).

(Botu3) V. Botu, R. Batra, J. Chapman and R. Ramprasad, https://arxiv.org/abs/1610.02098 (2016).

4.3 pair_style aip/water/2dm command

Accelerator Variant: *aip/water/2dm/opt*

4.3.1 Syntax

```
pair_style [hybrid/overlay ...] aip/water/2dm cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.3.2 Examples

```
pair_style  hybrid/overlay aip/water/2dm 16.0 1
pair_coeff  * * aip/water/2dm  CBNOH.aip.water.2dm C Ow Hw

pair_style  hybrid/overlay aip/water/2dm 16.0 lj/cut/tip4p/long 2 3 1 1 0.1546 10 8.5
pair_coeff  2 2  lj/cut/tip4p/long   8.0313e-3  3.1589     # O-O
pair_coeff  2 3  lj/cut/tip4p/long   0.0        0.0        # O-H
pair_coeff  3 3  lj/cut/tip4p/long   0.0        0.0        # H-H
pair_coeff  * *  aip/water/2dm       CBNOH.aip.water.2dm    C Ow Hw

pair_style  hybrid/overlay aip/water/2dm 16.0 lj/cut/tip4p/long 3 4 1 1 0.1546 10 8.5 coul/shield 16.0 1
pair_coeff  1*2 1*2  none
pair_coeff  3 3  lj/cut/tip4p/long   8.0313e-3  3.1589     # O-O
pair_coeff  3 4  lj/cut/tip4p/long   0.0        0.0        # O-H
pair_coeff  4 4  lj/cut/tip4p/long   0.0        0.0        # H-H
pair_coeff  * *  aip/water/2dm       CBNOH.aip.water.2dm  B N Ow Hw
pair_coeff  1 3  coul/shield         1.333
pair_coeff  1 4  coul/shield         1.333
pair_coeff  2 3  coul/shield         1.333
pair_coeff  2 4  coul/shield         1.333
```

4.3.3 Description

Added in version 15Jun2023.

The *aip/water/2dm* style computes the anisotropic interfacial potential (AIP) potential for interfaces of water with two-

dimensional (2D) materials as described in *(Feng1)* and *(Feng2)*.

$$
\begin{aligned}
E =& \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& \mathrm{Tap}(r_{ij}) \left\{ e^{-\alpha(r_{ij}/\beta - 1)} \left[\varepsilon + f(\rho_{ij}) + f(\rho_{ji})\right] - \frac{1}{1 + e^{-d\left[(r_{ij}/(s_R \cdot r^{eff})) - 1\right]}} \cdot \frac{C_6}{r_{ij}^6} \right\} \\
\rho_{ij}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_i)^2 \\
\rho_{ji}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_j)^2 \\
f(\rho) =& C e^{-(\rho/\delta)^2} \\
\mathrm{Tap}(r_{ij}) =& 20\left(\frac{r_{ij}}{R_{cut}}\right)^7 - 70\left(\frac{r_{ij}}{R_{cut}}\right)^6 + 84\left(\frac{r_{ij}}{R_{cut}}\right)^5 - 35\left(\frac{r_{ij}}{R_{cut}}\right)^4 + 1
\end{aligned}
$$

Where $\mathrm{Tap}(r_{ij})$ is the taper function which provides a continuous cutoff (up to third derivative) for interatomic separations larger than r_c *pair_style ilp_graphene_hbn*.

Note

This pair style uses the atomic normal vector definition from *(Feng1)*), where the atomic normal vectors of the hydrogen atoms are assumed to lie along the corresponding oxygen-hydrogen bonds and the normal vector of the central oxygen atom is defined as their average.

The provided parameter file, CBNOH.aip.water.2dm, is intended for use with *metal* *units*, with energies in meV. Two additional parameters, *S*, and *rcut* are included in the parameter file. *S* is designed to facilitate scaling of energies; *rcut* is the cutoff for an internal, short distance neighbor list that is generated for speeding up the calculation of the normals for all atom pairs.

Note

The parameters presented in the provided parameter file, CBNOH.aip.water.2dm, are fitted with the taper function enabled by setting the cutoff equal to 16.0 Angstrom. Using a different cutoff or taper function setting should be carefully checked as they can lead to significant errors. These parameters provide a good description in both short- and long-range interaction regimes. This is essential for simulations in high pressure regime (i.e., the interlayer distance is smaller than the equilibrium distance).

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using *pair_coeff* *settings* with the pair style set to *none*.

This pair style tallies a breakdown of the total interlayer potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 2. The 2 values correspond to the following sub-categories:

1. *E_vdW* = vdW (attractive) energy
2. *E_Rep* = Repulsive energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair aip/water/2dm
variable Evdw  equal c_0[1]
variable Erep  equal c_0[2]
thermo_style custom step temp epair v_Erep v_Evdw
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.3.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.3.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the newton setting to be *on* for pair interactions.

The CBNOH.aip.water.2dm potential file provided with LAMMPS is parameterized for *metal* units. You can use this pair style with any LAMMPS units, but you would need to create your own potential file with parameters in the appropriate units, if your simulation does not use *metal* units.

4.3.6 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style ilp_tmd*, *pair_style saip_metal*, *pair_style ilp_graphene_hbn*, *pair_style pair_kolmogorov_crespi_z*, *pair_style pair_kolmogorov_crespi_full*, *pair_style pair_lebedeva_z*, *pair_style pair_coul_shield*.

4.3.7 Default

tap_flag = 1

(Feng1) Z. Feng, . . . , and W. Ouyang, J. Phys. Chem. C. 127(18), 8704-8713 (2023).

(Feng2) Z. Feng, . . . , and W. Ouyang, Langmuir 39(50), 18198-18207 (2023).

4.4 pair_style airebo command

Accelerator Variants: *airebo/intel*, *airebo/omp*

4.5 pair_style airebo/morse command

Accelerator Variants: *airebo/morse/intel*, *airebo/morse/omp*

4.6 pair_style rebo command

Accelerator Variants: *rebo/intel*, *rebo/omp*

4.6.1 Syntax

```
pair_style style cutoff LJ_flag TORSION_flag cutoff_min
```

- style = *airebo* or *airebo/morse* or *rebo*
- cutoff = LJ or Morse cutoff (σ scale factor) (AIREBO and AIREBO-M only)
- LJ_flag = 0/1 to turn off/on the LJ or Morse term (AIREBO and AIREBO-M only, optional)
- TORSION_flag = 0/1 to turn off/on the torsion term (AIREBO and AIREBO-M only, optional)
- cutoff_min = Start of the transition region of cutoff (σ scale factor) (AIREBO and AIREBO-M only, optional)

4.6.2 Examples

```
pair_style airebo 3.0
pair_style airebo 2.5 1 0
pair_coeff * * ../potentials/CH.airebo H C

pair_style airebo/morse 3.0
pair_coeff * * ../potentials/CH.airebo-m H C

pair_style rebo
pair_coeff * * ../potentials/CH.rebo H C
```

4.6.3 Description

The *airebo* pair style computes the Adaptive Intermolecular Reactive Empirical Bond Order (AIREBO) Potential of *(Stuart)* for a system of carbon and/or hydrogen atoms. Note that this is the initial formulation of AIREBO from 2000, not the later formulation.

The *airebo/morse* pair style computes the AIREBO-M potential, which is equivalent to AIREBO, but replaces the LJ term with a Morse potential. The Morse potentials are parameterized by high-quality quantum chemistry (MP2) calculations and do not diverge as quickly as particle density increases. This allows AIREBO-M to retain accuracy to much higher pressures than AIREBO (up to 40 GPa for Polyethylene). Details for this potential and its parameterization are given in *(O'Conner)*.

The *rebo* pair style computes the Reactive Empirical Bond Order (REBO) Potential of *(Brenner)*. Note that this is the so-called second generation REBO from 2002, not the original REBO from 1990. As discussed below, second generation REBO is closely related to the initial AIREBO; it is just a subset of the potential energy terms with a few slightly different parameters

The AIREBO potential consists of three terms:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} \left[E_{ij}^{\mathrm{REBO}} + E_{ij}^{\mathrm{LJ}} + \sum_{k \neq i,j} \sum_{l \neq i,j,k} E_{kijl}^{\mathrm{TORSION}} \right]$$

By default, all three terms are included. For the *airebo* style, if the first two optional flag arguments to the pair_style command are included, the LJ and torsional terms can be turned off. Note that both or neither of the flags must be included. If both of the LJ an torsional terms are turned off, it becomes the second-generation REBO potential, with a small caveat on the spline fitting procedure mentioned below. This can be specified directly as pair_style *rebo* with no additional arguments.

The detailed formulas for this potential are given in *(Stuart)*; here we provide only a brief description.

The E^{REBO} term has the same functional form as the hydrocarbon REBO potential developed in *(Brenner)*. The coefficients for E^{REBO} in AIREBO are essentially the same as Brenner's potential, but a few fitted spline values are slightly different. For most cases the E^{REBO} term in AIREBO will produce the same energies, forces and statistical averages as the original REBO potential from which it was derived. The E^{REBO} term in the AIREBO potential gives the model its reactive capabilities and only describes short-ranged C-C, C-H and H-H interactions ($r < 2$Å). These interactions have strong coordination-dependence through a bond order parameter, which adjusts the attraction between the I,J atoms based on the position of other nearby atoms and thus has 3- and 4-body dependence.

The E^{LJ} term adds longer-ranged interactions ($2 < r <$ cutoff) using a form similar to the standard *Lennard Jones potential*. The E^{LJ} term in AIREBO contains a series of switching functions so that the short-ranged LJ repulsion ($1/r^{12}$) does not interfere with the energetics captured by the E^{REBO} term. The extent of the E^{LJ} interactions is determined by the *cutoff* argument to the pair_style command which is a scale factor. For each type pair (C-C, C-H, H-H) the cutoff is obtained by multiplying the scale factor by the sigma value defined in the potential file for that type pair. In the standard AIREBO potential, $\sigma_{CC} = 3.4$Å, so with a scale factor of 3.0 (the argument in pair_style), the resulting E^{LJ} cutoff would be 10.2Å.

By default, the longer-ranged interaction is smoothly switched off between 2.16 and 3.0 σ. By specifying *cutoff_min* in addition to *cutoff*, the switching can be configured to take place between *cutoff_min* and *cutoff*. *cutoff_min* can only be specified if all optional arguments are given.

The E^{TORSION} term is an explicit 4-body potential that describes various dihedral angle preferences in hydrocarbon configurations.

Only a single pair_coeff command is used with the *airebo*, *airebo* or *rebo* style which specifies an AIREBO, REBO, or AIREBO-M potential file with parameters for C and H. Note that as of LAMMPS version 15 May 2019 the *rebo* style in LAMMPS uses its own potential file (CH.rebo). These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- *N* element names = mapping of AIREBO elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, if your LAMMPS simulation has 4 atom types and you want the first 3 to be C, and the fourth to be H, you would use the following pair_coeff command:

```
pair_coeff * * CH.airebo C C C H
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three C arguments map LAMMPS atom types 1,2,3 to the C element in the AIREBO file. The final H argument maps LAMMPS atom type 4 to the H element in the AIREBO file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *airebo* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The parameters/coefficients for the AIREBO potentials are listed in the CH.airebo file to agree with the original *(Stuart)* paper. Thus the parameters are specific to this potential and the way it was fit, so modifying the file should be done cautiously.

Similarly the parameters/coefficients for the AIREBO-M potentials are listed in the CH.airebo-m file to agree with the *(O'Connor)* paper. Thus the parameters are specific to this potential and the way it was fit, so modifying the file should be done cautiously. The AIREBO-M Morse potentials were parameterized using a cutoff of 3.0 (σ). Modifying this cutoff may impact simulation accuracy.

This pair style tallies a breakdown of the total AIREBO potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 3. The 3 values correspond to the following sub-categories:

1. E_{REBO} = REBO energy
2. E_{LJ} = Lennard-Jones energy
3. E_{TORSION} = Torsion energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair airebo
variable REBO     equal c_0[1]
variable LJ       equal c_0[2]
variable TORSION  equal c_0[3]
thermo_style custom step temp epair v_REBO v_LJ v_TORSION
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.6.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support the *pair_modify* mix, shift, table, and tail options.

These pair styles do not write their information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.6.5 Restrictions

These pair styles are part of the MANYBODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair potentials require the *newton* setting to be "on" for pair interactions.

The CH.airebo and CH.airebo-m potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the pair styles with *any* LAMMPS units, but you would need to create your own AIREBO or AIREBO-M potential file with coefficients listed in the appropriate units, if your simulation does not use "metal" units.

The pair styles provided here **only** support potential files parameterized for the elements carbon and hydrogen (designated with "C" and "H" in the *pair_coeff* command. Using potential files for other elements will trigger an error.

4.6.6 Related commands

pair_coeff

4.6.7 Default

none

(Stuart) Stuart, Tutein, Harrison, J Chem Phys, 112, 6472-6486 (2000).

(Brenner) Brenner, Shenderova, Harrison, Stuart, Ni, Sinnott, J Physics: Condensed Matter, 14, 783-802 (2002).

(O'Connor) O'Connor et al., J. Chem. Phys. 142, 024903 (2015).

4.7 pair_style amoeba command

Accelerator Variants: *amoeba/gpu*

4.8 pair_style hippo command

Accelerator Variants: *hippo/gpu*

4.8.1 Syntax

```
pair_style style
```

- style = *amoeba* or *hippo*

4.8.2 Examples

```
pair_style amoeba
pair_coeff * * protein.prm.amoeba protein.key.amoeba
```

```
pair_style hippo
pair_coeff * * water.prm.hippo water.key.hippo
```

4.8.3 Additional info

- *Howto amoeba*
- examples/amoeba
- tools/amoeba
- potentials/*.amoeba
- potentials/*.hippo

4.8.4 Description

The *amoeba* style computes the AMOEBA polarizable field formulated by Jay Ponder's group at the U Washington at St Louis *(Ren)*, *(Shi)*. The *hippo* style computes the HIPPO polarizable force field, an extension to AMOEBA, formulated by Josh Rackers and collaborators in the Ponder group *(Rackers)*.

These force fields can be used when polarization effects are desired in simulations of water, organic molecules, and biomolecules including proteins, provided that parameterizations (Tinker PRM force field files) are available for the systems you are interested in. Files in the LAMMPS potentials directory with a "amoeba" or "hippo" suffix can be used. The Tinker distribution and website have additional force field files as well.

As discussed on the *Howto amoeba* doc page, the intermolecular (non-bonded) portion of the AMOEBA force field contains these terms:

$$U_{amoeba} = U_{multipole} + U_{polar} + U_{hal}$$

while the HIPPO force field contains these terms:

$$U_{hippo} = U_{multipole} + U_{polar} + U_{qxfer} + U_{repulsion} + U_{dispersion}$$

Conceptually, these terms compute the following interactions:

- U_{hal} = buffered 14-7 van der Waals with offsets applied to hydrogen atoms
- $U_{repulsion}$ = Pauli repulsion due to rearrangement of electron density
- $U_{dispersion}$ = dispersion between correlated, instantaneous induced dipole moments
- $U_{multipole}$ = electrostatics between permanent point charges, dipoles, and quadrupoles
- U_{polar} = electronic polarization between induced point dipoles
- U_{qxfer} = charge transfer effects

Note that the AMOEBA versus HIPPO force fields typically compute the same term differently using their own formulas. The references on this doc page give full details for both force fields.

The formulas for the AMOEBA energy terms are:

$$
\begin{aligned}
U_{hal} =& \varepsilon_{ij} \left(\frac{1.07}{\rho_{ij} + 0.07} \right)^7 \left(\frac{1.12}{\rho_{ij}^7 + 0.12} - 2 \right) \\
U_{multipole} =& \vec{M}_i T_{ij} \vec{M}_j, \quad \text{with} \quad \vec{M} = \left(q, \vec{\mu}_{perm}, \Theta \right) \\
U_{polar} =& \frac{1}{2} \vec{\mu}_i^{ind} \vec{E}_i^{perm}
\end{aligned}
$$

The formulas for the HIPPO energy terms are:

$$
\begin{aligned}
U_{multipole} =& Z_i \frac{1}{r_{ij}} Z_j + Z_i T_{ij}^{damp} \vec{M}_j + Z_j T_{ji}^{damp} \vec{M}_i + \vec{M}_i T_{ij}^{damp} \vec{M}_j, \quad \text{with} \quad \vec{M} = \left(q, \vec{\mu}_{perm}, \Theta \right) \\
U_{polar} =& \frac{1}{2} \vec{\mu}_i^{ind} \vec{E}_i^{perm} \\
U_{qxfer} =& \varepsilon_i e^{-\eta_j r_{ij}} + \varepsilon_j e^{-\eta_i r_{ij}} \\
U_{repulsion} =& \frac{K_i K_j}{r_{ij}} S^2 S^2 = \left(\int \phi_i \phi_j dv \right)^2 = \vec{M}_i T_{ij}^{repulsion} \vec{M}_j \\
U_{dispersion} =& - \frac{C_6^i C_6^j}{r_{ij}^6} \left(f_{damp}^{dispersion} \right)_{ij}^2
\end{aligned}
$$

Note

The AMOEBA and HIPPO force fields compute long-range charge, dipole, and quadrupole interactions as well as long-range dispersion effects. However, unlike other models with long-range interactions in LAMMPS, this does not require use of a KSpace style via the *kspace_style* command. That is because for AMOEBA and HIPPO the long-range computations are intertwined with the pairwise computations. So these pair style include both short- and long-range computations. This means the energy and virial computed by the pair style as well as the "Pair" timing reported by LAMMPS will include the long-range calculations.

The implementation of the AMOEBA and HIPPO force fields in LAMMPS was done using F90 code provided by the Ponder group from their Tinker MD code.

The current implementation (July 2022) of AMOEBA in LAMMPS matches the version discussed in *(Ponder)*, *(Ren)*, and *(Shi)*. Likewise the current implementation of HIPPO in LAMMPS matches the version discussed in *(Rackers)*.

Added in version 8Feb2023.

Accelerator support via the GPU package is available.

Only a single pair_coeff command is used with either the *amoeba* and *hippo* styles which specifies two Tinker files, a PRM and KEY file.

```
pair_coeff * * ../potentials/protein.prm.amoeba ../potentials/protein.key.amoeba
pair_coeff * * ../potentials/water.prm.hippo ../potentials/water.key.hippo
```

Examples of the PRM files are in the potentials directory with an *.amoeba or *.hippo suffix. The examples/amoeba directory has examples of both PRM and KEY files.

A Tinker PRM file is composed of sections, each of which has multiple lines. A Tinker KEY file is composed of lines, each of which has a keyword followed by zero or more parameters.

The list of PRM sections and KEY keywords which LAMMPS recognizes are listed on the *Howto amoeba* doc page. If not recognized, the section or keyword is skipped.

Note that if the KEY file is specified as NULL, then no file is required; default values for various AMOEBA/HIPPO settings are used. The *Howto amoeba* doc page also gives the default settings.

Added in version 3Nov2022.

The *amoeba* and *hippo* pair styles support extraction of two per-atom quantities by the *fix pair* command. This allows the quantities to be output to files by the *dump* or otherwise processed by other LAMMPS commands.

The names of the two quantities are "uind" and "uinp" for the induced dipole moments for each atom. Neither quantity needs to be triggered by the *fix pair* command in order for these pair styles to calculate it.

4.8.5 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support the *pair_modify* mix, shift, table, and tail options.

These pair styles do not write their information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

Note

Using the GPU accelerated pair styles 'amoeba/gpu' or 'hippo/gpu' when compiling the GPU package for OpenCL has a few known issues when running on integrated GPUs and the calculation may crash.

The GPU accelerated pair styles are also not (yet) compatible with single precision FFTs.

4.8.6 Restrictions

These pair styles are part of the AMOEBA package. They are only enabled if LAMMPS was built with that package. See the *Build package* doc page for more info.

The AMOEBA and HIPPO potential (PRM) and KEY files provided with LAMMPS in the potentials and examples/amoeba directories are Tinker files parameterized for Tinker units. Their numeric parameters are converted by LAMMPS to its real units *units*. Thus you can only use these pair styles with real units.

These potentials do not yet calculate per-atom energy or virial contributions.

As explained on the *AMOEBA and HIPPO howto* page, use of these pair styles to run a simulation with the AMOEBA or HIPPO force fields requires several things.

The first is a data file generated by the tools/tinker/tinker2lmp.py conversion script which uses Tinker file force field file input to create a data file compatible with LAMMPS.

The second is use of these commands:

- *atom_style amoeba*
- *fix property/atom*
- *special_bonds one/five*

And third, depending on the model being simulated, these commands for intramolecular interactions may also be required:

- *bond_style class2*
- *angle_style amoeba*
- *dihedral_style fourier*
- *improper_style amoeba*
- *fix amoeba/pitorsion*
- *fix amoeba/bitorsion*

4.8.7 Related commands

atom_style amoeba, *bond_style class2*, *angle_style amoeba*, *dihedral_style fourier*, *improper_style amoeba*, *fix amoeba/pitorsion*, *fix amoeba/bitorsion*, *special_bonds one/five*, *fix property/atom*

4.8.8 Default

none

(Ponder) Ponder, Wu, Ren, Pande, Chodera, Schnieders, Haque, Mobley, Lambrecht, DiStasio Jr, M. Head-Gordon, Clark, Johnson, T. Head-Gordon, J Phys Chem B, 114, 2549-2564 (2010).

(Rackers) Rackers, Silva, Wang, Ponder, J Chem Theory Comput, 17, 7056-7084 (2021).

(Ren) Ren and Ponder, J Phys Chem B, 107, 5933 (2003).

(Shi) Shi, Xia, Zhang, Best, Wu, Ponder, Ren, J Chem Theory Comp, 9, 4046, 2013.

4.9 pair_style atm command

4.9.1 Syntax

```
pair_style atm cutoff cutoff_triple
```

- cutoff = cutoff for each pair in 3-body interaction (distance units)
- cutoff_triple = additional cutoff applied to product of 3 pairwise distances (distance units)

4.9.2 Examples

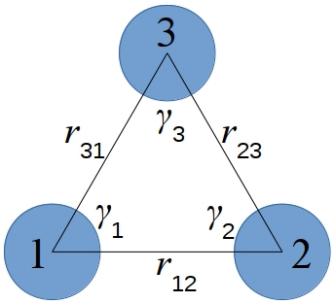
```
pair_style atm 4.5 2.5
pair_coeff * * * 0.072

pair_style hybrid/overlay lj/cut 6.5 atm 4.5 2.5
pair_coeff * * lj/cut 1.0 1.0
pair_coeff 1 1 atm 1 0.064
pair_coeff 1 1 atm 2 0.080
pair_coeff 1 2 atm 2 0.100
pair_coeff 2 2 atm 2 0.125
```

4.9.3 Description

The *atm* style computes a 3-body *Axilrod-Teller-Muto* potential for the energy E of a system of atoms as

$$E = \nu \frac{1 + 3\cos\gamma_1 \cos\gamma_2 \cos\gamma_3}{r_{12}^3 r_{23}^3 r_{31}^3}$$

where ν is the three-body interaction strength. The distances between pairs of atoms r_{12}, r_{23}, r_{31} and the angles γ_1, γ_2, γ_3 are as shown in this diagram:



Note that for the interaction between a triplet of atoms I, J, K, there is no "central" atom. The interaction is symmetric with respect to permutation of the three atoms. Thus the ν value is the same for all those permutations of the atom types of I, J, K and needs to be specified only once, as discussed below.

The *atm* potential is typically used in combination with a two-body potential using the *pair_style hybrid/overlay* command as in the example above.

The potential for a triplet of atom is calculated only if all 3 distances r_{12}, r_{23}, r_{31} between the three atoms satisfy $r_{IJ} <$ cutoff. In addition, the product of the 3 distances $r_{12} r_{23} r_{31} <$ cutoff_triple ^{3} is required, which excludes from calculation the triplets with small contribution to the interaction.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the restart files read by the *read_restart* commands:

- K = atom type of the third atom (1 to N_{types})
- ν = prefactor (energy/distance^9 units)

K can be specified in one of two ways. An explicit numeric value or type label can be used, as in the second example above. LAMMPS sets the coefficients for the other 5 symmetric interactions to the same values. E.g. if $I = 1$, $J = 2$, $K = 3$, then these 6 values are set to the specified ν: ν_{123}, ν_{132}, ν_{213}, ν_{231}, ν_{312}, ν_{321}. This enforces the symmetry discussed above.

A wildcard asterisk can be used for K to set the coefficients for multiple triplets of atom types. This takes the form "*" or "*n" or "n*" or "m*n". If N equals the number of atom types, then an asterisk with no numeric values means all types from 1 to N. A leading asterisk means all types from 1 to n (inclusive). A trailing asterisk means all types from n to N (inclusive). A middle asterisk means all types from m to n (inclusive). Note that only type triplets with $J \leq K$ are considered; if asterisks imply type triplets where $K < J$, they are ignored.

Note that a pair_coeff command can override a previous setting for the same I, J, K triplet. For example, these commands set ν for all $I, J.K$ triplets, then overwrite nu for just the $I, J, K = 2, 3, 4$ triplet:

```
pair_coeff * * * 0.25
pair_coeff 2 3 4 0.1
```

Note that for a simulation with a single atom type, only a single entry is required, e.g.

```
pair_coeff 1 1 1 0.25
```

For a simulation with two atom types, four pair_coeff commands will specify all possible nu values:

```
pair_coeff 1 1 1 nu1
pair_coeff 1 1 2 nu2
pair_coeff 1 2 2 nu3
pair_coeff 2 2 2 nu4
```

For a simulation with three atom types, ten pair_coeff commands will specify all possible nu values:

```
pair_coeff 1 1 1 nu1
pair_coeff 1 1 2 nu2
pair_coeff 1 1 3 nu3
pair_coeff 1 2 2 nu4
pair_coeff 1 2 3 nu5
pair_coeff 1 3 3 nu6
pair_coeff 2 2 2 nu7
pair_coeff 2 2 3 nu8
pair_coeff 2 3 3 nu9
pair_coeff 3 3 3 nu10
```

By default the ν value for all triplets is set to 0.0. Thus it is not required to provide pair_coeff commands that enumerate triplet interactions for all K types. If some I, J, K combination is not specified, then there will be no 3-body ATM interactions for that combination and all its permutations. However, as with all pair styles, it is required to specify a pair_coeff command for all I, J combinations, else an error will result.

4.9.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style do not support the *pair_modify* mix, shift, table, and tail options.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file. However, if the *atm* potential is used in combination with other potentials using the *pair_style hybrid/overlay* command then pair_coeff commands need to be re-specified in the restart input script.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, and *outer* keywords.

4.9.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.9.6 Related commands

pair_coeff

4.9.7 Default

none

(Axilrod) Axilrod and Teller, J Chem Phys, 11, 299 (1943); Muto, Nippon Sugaku-Buturigakkwaishi 17, 629 (1943).

4.10 pair_style awpmd/cut command

4.10.1 Syntax

```
pair_style awpmd/cut Rc keyword value ...
```

- Rc = global cutoff, -1 means cutoff of half the shortest box length
- zero or more keyword/value pairs may be appended
- keyword = *hartree* or *dproduct* or *uhf* or *free* or *pbc* or *fix* or *harm* or *ermscale* or *flex_press*

```
hartree value = none
dproduct value = none
uhf value = none
free value = none
pbc value = Plen
  Plen = periodic width of electron = -1 or positive value (distance units)
fix value = Flen
  Flen = fixed width of electron = -1 or positive value (distance units)
harm value = width
  width = harmonic width constraint
```

ermscale value = factor
  factor = scaling between electron mass and width variable mass
flex_press value = none

4.10.2 Examples

```
pair_style awpmd/cut -1
pair_style awpmd/cut 40.0 uhf free
pair_coeff * *
pair_coeff 2 2 20.0
```

4.10.3 Description

This pair style contains an implementation of the Antisymmetrized Wave Packet Molecular Dynamics (AWPMD) method. Need citation here. Need basic formulas here. Could be links to other documents.

Rc is the cutoff.

The pair_style command allows for several optional keywords to be specified.

The *hartree*, *dproduct*, and *uhf* keywords specify the form of the initial trial wave function for the system. If the *hartree* keyword is used, then a Hartree multielectron trial wave function is used. If the *dproduct* keyword is used, then a trial function which is a product of two determinants for each spin type is used. If the *uhf* keyword is used, then an unrestricted Hartree-Fock trial wave function is used.

The *free*, *pbc*, and *fix* keywords specify a width constraint on the electron wave packets. If the *free* keyword is specified, then there is no constraint. If the *pbc* keyword is used and *Plen* is specified as -1, then the maximum width is half the shortest box length. If *Plen* is a positive value, then the value is the maximum width. If the *fix* keyword is used and *Flen* is specified as -1, then electrons have a constant width that is read from the data file. If *Flen* is a positive value, then the constant width for all electrons is set to *Flen*.

The *harm* keyword allow oscillations in the width of the electron wave packets. More details are needed.

The *ermscale* keyword specifies a unitless scaling factor between the electron masses and the width variable mass. More details needed.

If the *flex_press* keyword is used, then a contribution from the electrons is added to the total virial and pressure of the system.

This potential is designed to be used with *atom_style wavepacket* definitions, in order to handle the description of systems with interacting nuclei and explicit electrons.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutoff (distance units)

For *awpmd/cut*, the cutoff coefficient is optional. If it is not used (as in some of the examples above), the default global value specified in the pair_style command is used.

4.10.4 Mixing, shift, table, tail correction, restart, rRESPA info

The *pair_modify* mix, shift, table, and tail options are not relevant for this pair style.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.10.5 Restrictions

none

4.10.6 Related commands

pair_coeff

4.10.7 Default

These are the defaults for the pair_style keywords: *hartree* for the initial wave function, *free* for the wave packet width.

4.11 pair_style beck command

Accelerator Variants: *beck/gpu*, *beck/omp*

4.11.1 Syntax

```
pair_style beck Rc
```

- Rc = cutoff for interactions (distance units)

4.11.2 Examples

```
pair_style beck 8.0
pair_coeff * * 399.671876712 0.0000867636112694 0.675 4.390 0.0003746
pair_coeff 1 1 399.671876712 0.0000867636112694 0.675 4.390 0.0003746 6.0
```

4.11.3 Description

Style *beck* computes interactions based on the potential by *(Beck)*, originally designed for simulation of Helium. It includes truncation at a cutoff distance r_c.

$$E(r) = A \exp\left[-\alpha r - \beta r^6\right] - \frac{B}{\left(r^2 + a^2\right)^3}\left(1 + \frac{2.709 + 3a^2}{r^2 + a^2}\right) \qquad r < r_c$$

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands.

- A (energy units)
- B (energy-distance^6 units)
- a (distance units)
- α (1/distance units)
- β (1/distance^6 units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff r_c is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.11.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, coefficients must be specified. No default mixing rules are used.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.11.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.11.6 Related commands

pair_coeff

4.11.7 Default

none

(Beck) Beck, Molecular Physics, 14, 311 (1968).

4.12 pair_style body/nparticle command

4.12.1 Syntax

```
pair_style body/nparticle cutoff
```

cutoff = global cutoff for interactions (distance units)

4.12.2 Examples

```
pair_style body/nparticle 3.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.0 1.5 2.5
```

4.12.3 Description

Style *body/nparticle* is for use with body particles and calculates pairwise body/body interactions as well as interactions between body and point-particles. See the *Howto body* doc page for more details on using body particles.

This pair style is designed for use with the "nparticle" body style, which is specified as an argument to the "atom-style body" command. See the *Howto body* page for more details about the body styles LAMMPS supports. The "nparticle" style treats a body particle as a rigid body composed of N sub-particles.

The coordinates of a body particle are its center-of-mass (COM). If the COMs of a pair of body particles are within the cutoff (global or type-specific, as specified above), then all interactions between pairs of sub-particles in the two body particles are computed. E.g. if the first body particle has 3 sub-particles, and the second has 10, then 30 interactions are computed and summed to yield the total force and torque on each body particle.

Note

In the example just described, all 30 interactions are computed even if the distance between a particular pair of sub-particles is greater than the cutoff. Likewise, no interaction between two body particles is computed if the two COMs are further apart than the cutoff, even if the distance between some pairs of their sub-particles is within the cutoff. Thus care should be used in defining the cutoff distances for body particles, depending on their shape and size.

Similar rules apply for a body particle interacting with a point particle. The distance between the two particles is calculated using the COM of the body particle and the position of the point particle. If the distance is within the cutoff and the body particle has N sub-particles, then N interactions with the point particle are computed and summed. If the distance is not within the cutoff, no interactions between the body and point particle are computed.

The interaction between two sub-particles, or a sub-particle and point particle, or between two point particles is computed as a Lennard-Jones interaction, using the standard formula

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < R_c$$

where R_c is the cutoff. As explained above, an interaction involving one or two body sub-particles may be computed even for $r > R_c$.

For style *body*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- ε (energy units)
- σ (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

4.12.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of this pair style can be mixed. The default mix value is *geometric*. See the *pair_modify* command for details.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.12.5 Restrictions

This style is part of the BODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Defining particles to be bodies so they participate in body/body or body/particle interactions requires the use of the *atom_style body* command.

4.12.6 Related commands

pair_coeff, *fix rigid*

4.12.7 Default

none

4.13 pair_style body/rounded/polygon command

4.13.1 Syntax

```
pair_style body/rounded/polygon c_n c_t mu delta_ua cutoff
```

```
c_n = normal damping coefficient
c_t = tangential damping coefficient
mu = normal friction coefficient during gross sliding
delta_ua = multiple contact scaling factor
cutoff = global separation cutoff for interactions (distance units), see below for definition
```

4.13.2 Examples

```
pair_style body/rounded/polygon 20.0 5.0 0.0 1.0 0.5
pair_coeff * * 100.0 1.0
pair_coeff 1 1 100.0 1.0
```

4.13.3 Description

Style *body/rounded/polygon* is for use with 2d models of body particles of style *rounded/polygon*. It calculates pairwise body/body interactions which can include body particles modeled as 1-vertex circular disks with a specified diameter. See the *Howto body* page for more details on using body rounded/polygon particles.

This pairwise interaction between rounded polygons is described in *Fraige*, where a polygon does not have sharp corners, but is rounded at its vertices by circles centered on each vertex with a specified diameter. The edges of the polygon are defined between pairs of adjacent vertices. The circle diameter for each polygon is specified in the data file read by the *read data* command. This is a 2d discrete element model (DEM) which allows for multiple contact points.

Note that when two particles interact, the effective surface of each polygon particle is displaced outward from each of its vertices and edges by half its circle diameter (as in the diagram below of a gray and yellow square particle).

The interaction forces and energies between two particles are defined with respect to the separation of their respective rounded surfaces, not by the separation of the vertices and edges themselves.

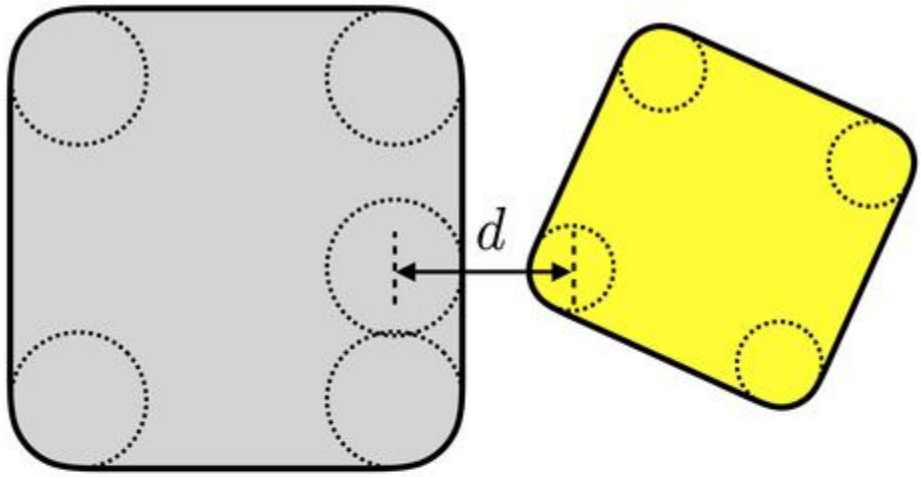
This means that the specified cutoff in the pair_style command is the cutoff distance, r_c, for the surface separation, δ_n (see figure below). This is the distance at which two particles no longer interact. If r_c is specified as 0.0, then it is a contact-only interaction. I.e. the two particles must overlap in order to exert a repulsive force on each other. If $r_c > 0.0$, then the force between two particles will be attractive for surface separations from 0 to r_c, and repulsive once the particles overlap.

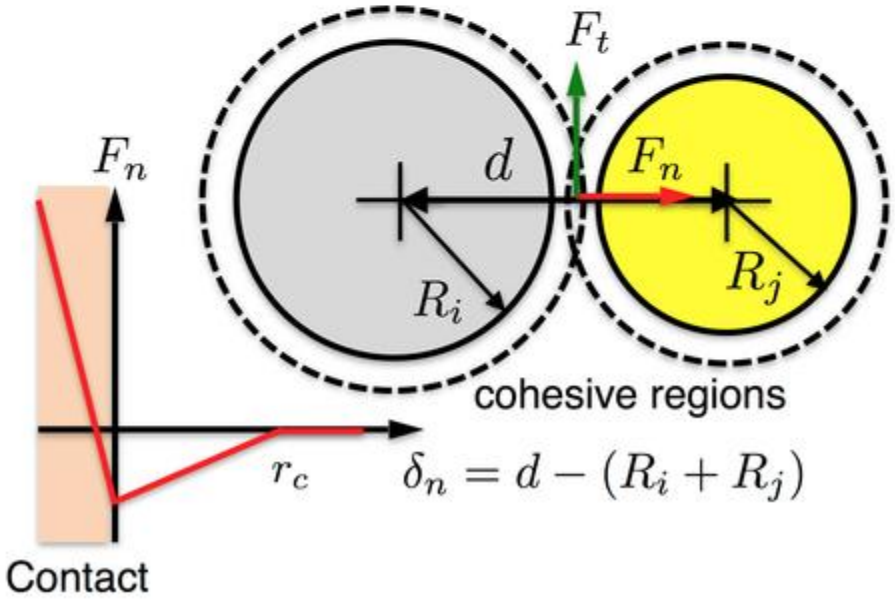
Note that unlike for other pair styles, the specified cutoff is not the distance between the centers of two particles at which they stop interacting. This center-to-center distance depends on the shape and size of the two particles and their relative orientation. LAMMPS takes that into account when computing the surface separation distance and applying the r_c cutoff.

The forces between vertex-vertex, vertex-edge, and edge-edge overlaps are given by:

$$F_n = \begin{cases} k_n \delta_n - c_n v_n & \delta_n \leq 0 \\ -k_{na} \delta_n - c_n v_n & 0 < \delta_n \leq r_c \\ 0 & \delta_n > r_c \end{cases}$$

$$F_t = \begin{cases} \mu k_n \delta_n - c_t v_t & \delta_n \leq 0 \\ 0 & \delta_n > 0 \end{cases}$$





Note that F_n and F_t are functions of the surface separation $\delta_n = d - (R_i + R_j)$. In this model, when $(R_i + R_j) < d < (R_i + R_j) + r_c$, that is, $0 < \delta_n < r_c$, the cohesive region of the two surfaces overlap and the two surfaces are attractive to each other.

In *Fraige*, the tangential friction force between two particles that are in contact is modeled differently prior to gross sliding (i.e. static friction) and during gross-sliding (kinetic friction). The latter takes place when the tangential deformation exceeds the Coulomb frictional limit. In the current implementation, however, we do not take into account frictional history, i.e. we do not keep track of how many time steps the two particles have been in contact nor calculate the tangential deformation. Instead, we assume that gross sliding takes place as soon as two particles are in contact.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file read by the *read_data* command:

- k_n (energy/distance^2 units)
- k_{na} (energy/distance^2 units)

Effectively, k_n and k_{na} are the slopes of the red lines in the plot above for force versus surface separation, for $\delta_n < 0$ and $0 < \delta_n < r_c$ respectively.

4.13.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.13.5 Restrictions

These pair styles are part of the BODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

4.13.6 Related commands

pair_coeff

4.13.7 Default

none

(Fraige) F. Y. Fraige, P. A. Langston, A. J. Matchett, J. Dodds, Particuology, 6, 455 (2008).

4.14 pair_style body/rounded/polyhedron command

4.14.1 Syntax

```
pair_style body/rounded/polyhedron c_n c_t mu delta_ua cutoff
```

```
c_n = normal damping coefficient
c_t = tangential damping coefficient
mu = normal friction coefficient during gross sliding
delta_ua = multiple contact scaling factor
cutoff = global separation cutoff for interactions (distance units), see below for definition
```

4.14.2 Examples

```
pair_style body/rounded/polyhedron 20.0 5.0 0.0 1.0 0.5
pair_coeff * * 100.0 1.0
pair_coeff 1 1 100.0 1.0
```

4.14.3 Description

Style *body/rounded/polygon* is for use with 3d models of body particles of style *rounded/polyhedron*. It calculates pairwise body/body interactions which can include body particles modeled as 1-vertex spheres with a specified diameter. See the *Howto body* page for more details on using body rounded/polyhedron particles.

This pairwise interaction between the rounded polyhedra is described in *Wang*, where a polyhedron does not have sharp corners and edges, but is rounded at its vertices and edges by spheres centered on each vertex with a specified diameter. The edges of the polyhedron are defined between pairs of adjacent vertices. Its faces are defined by a loop of edges. The sphere diameter for each polygon is specified in the data file read by the *read data* command. This is a discrete element model (DEM) which allows for multiple contact points.

Note that when two particles interact, the effective surface of each polyhedron particle is displaced outward from each of its vertices, edges, and faces by half its sphere diameter. The interaction forces and energies between two particles are defined with respect to the separation of their respective rounded surfaces, not by the separation of the vertices, edges, and faces themselves.

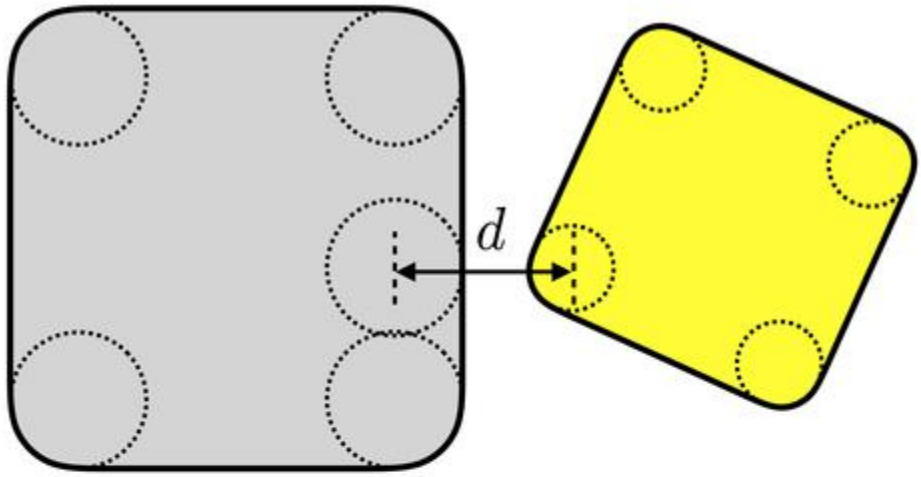
This means that the specified cutoff in the pair_style command is the cutoff distance, r_c, for the surface separation, δ_n (see figure below). This is the distance at which two particles no longer interact. If r_c is specified as 0.0, then it is a contact-only interaction. I.e. the two particles must overlap in order to exert a repulsive force on each other. If $r_c > 0.0$, then the force between two particles will be attractive for surface separations from 0 to r_c, and repulsive once the particles overlap.

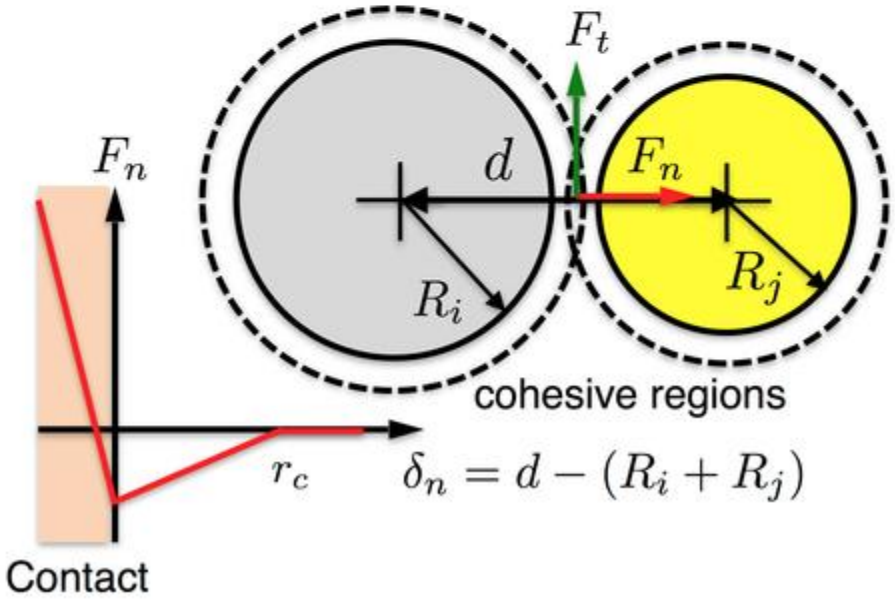
Note that unlike for other pair styles, the specified cutoff is not the distance between the centers of two particles at which they stop interacting. This center-to-center distance depends on the shape and size of the two particles and their relative orientation. LAMMPS takes that into account when computing the surface separation distance and applying the r_c cutoff.

The forces between vertex-vertex, vertex-edge, vertex-face, edge-edge, and edge-face overlaps are given by:

$$F_n = \begin{cases} k_n\delta_n - c_n v_n, & \delta_n \le 0 \\ -k_{na}\delta_n - c_n v_n & 0 < \delta_n \le r_c \\ 0 & \delta_n > r_c \end{cases}$$

$$F_t = \begin{cases} \mu k_n \delta_n - c_t v_t & \delta_n \le 0 \\ 0 & \delta_n > 0 \end{cases}$$





In *Wang*, the tangential friction force between two particles that are in contact is modeled differently prior to gross sliding (i.e. static friction) and during gross-sliding (kinetic friction). The latter takes place when the tangential deformation exceeds the Coulomb frictional limit. In the current implementation, however, we do not take into account frictional history, i.e. we do not keep track of how many time steps the two particles have been in contact nor calculate the tangential deformation. Instead, we assume that gross sliding takes place as soon as two particles are in contact.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file read by the *read_data* command:

- k_n (energy/distance^2 units)
- k_{na} (energy/distance^2 units)

Effectively, k_n and k_{na} are the slopes of the red lines in the plot above for force versus surface separation, for δ_n < 0 and $0 < \delta_n < r_c$ respectively.

4.14.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.14.5 Restrictions

These pair styles are part of the BODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

4.14.6 Related commands

pair_coeff

4.14.7 Default

none

(Wang) J. Wang, H. S. Yu, P. A. Langston, F. Y. Fraige, Granular Matter, 13, 1 (2011).

4.15 pair_style bop command

4.15.1 Syntax

```
pair_style bop keyword ...
```

- zero or more keywords may be appended
- keyword = *save*

```
save = pre-compute and save some values
```

4.15.2 Examples

```
pair_style bop
pair_coeff * * ../potentials/CdTe_bop Cd Te
pair_style bop save
pair_coeff * * ../potentials/CdTe.bop.table Cd Te Te
comm_modify cutoff 14.70
```

4.15.3 Description

The *bop* pair style computes Bond-Order Potentials (BOP) based on quantum mechanical theory incorporating both σ and π bonding. By analytically deriving the BOP from quantum mechanical theory its transferability to different phases can approach that of quantum mechanical methods. This potential is similar to the original BOP developed by Pettifor (*Pettifor_1*, *Pettifor_2*, *Pettifor_3*) and later updated by Murdick, Zhou, and Ward (*Murdick*, *Ward*). Currently, BOP potential files for these systems are provided with LAMMPS: AlCu, CCu, CdTe, CdTeSe, CdZnTe, CuH, GaAs. A system with only a subset of these elements, including a single element (e.g. C or Cu or Al or Ga or Zn or CdZn), can also be modeled by using the appropriate alloy file and assigning all atom types to the single element or subset of elements via the pair_coeff command, as discussed below.

The BOP potential consists of three terms:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=i_1}^{i_N}\phi_{ij}\left(r_{ij}\right) - \sum_{i=1}^{N}\sum_{j=i_1}^{i_N}\beta_{\sigma,ij}\left(r_{ij}\right)\cdot\Theta_{\sigma,ij} - \sum_{i=1}^{N}\sum_{j=i_1}^{i_N}\beta_{\pi,ij}\left(r_{ij}\right)\cdot\Theta_{\pi,ij} + U_{prom}$$

where $\phi_{ij}(r_{ij})$ is a short-range two-body function representing the repulsion between a pair of ion cores, $\beta_{\sigma,ij}(r_{ij})$ and $\beta_{\sigma,ij}(r_{ij})$ are respectively sigma and π bond integrals, $\Theta_{\sigma,ij}$ and $\Theta_{\pi,ij}$ are σ and π bond-orders, and U_prom is the promotion energy for sp-valent systems.

The detailed formulas for this potential are given in Ward (*Ward*); here we provide only a brief description.

The repulsive energy $\phi_{ij}(r_{ij})$ and the bond integrals $\beta_{\sigma,ij}(r_{ij})$ and $\beta_{\phi,ij}(r_{ij})$ are functions of the interatomic distance r_{ij} between atom i and j. Each of these potentials has a smooth cutoff at a radius of $r_{cut,ij}$. These smooth cutoffs ensure stable behavior at situations with high sampling near the cutoff such as melts and surfaces.

The bond-orders can be viewed as environment-dependent local variables that are ij bond specific. The maximum value of the σ bond-order (Θ_σ is 1, while that of the π bond-order (Θ_π) is 2, attributing to a maximum value of the total bond-order (Θ_σ + Θ_π) of 3. The σ and π bond-orders reflect the ubiquitous single-, double-, and triple- bond behavior of chemistry. Their analytical expressions can be derived from tight- binding theory by recursively expanding an inter-site Green's function as a continued fraction. To accurately represent the bonding with a computationally efficient potential formulation suitable for MD simulations, the derived BOP only takes (and retains) the first two levels of the recursive representations for both the σ and the π bond-orders. Bond-order terms can be understood in terms of molecular orbital hopping paths based upon the Cyrot-Lackmann theorem (*Pettifor_1*). The σ bond-order with a half-full valence shell is used to interpolate the bond-order expression that incorporated explicit valance band filling. This π bond-order expression also contains also contains a three-member ring term that allows implementation of an asymmetric density of states, which helps to either stabilize or destabilize close-packed structures. The π bond-order includes hopping paths of length 4. This enables the incorporation of dihedral angles effects.

Note

Note that unlike for other potentials, cutoffs for BOP potentials are not set in the pair_style or pair_coeff command; they are specified in the BOP potential files themselves. Likewise, the BOP potential files list atomic masses; thus you do not need to use the *mass* command to specify them. Note that for BOP potentials with hydrogen, you will likely want to set the mass of H atoms to be 10x or 20x larger to avoid having to use a tiny timestep. You can do this by using the *mass* command after using the *pair_coeff* command to read the BOP potential file.

One option can be specified as a keyword with the pair_style command.

The *save* keyword gives you the option to calculate in advance and store a set of distances, angles, and derivatives of angles. The default is to not do this, but to calculate them on-the-fly each time they are needed. The former may be faster, but takes more memory. The latter requires less memory, but may be slower. It is best to test this option to optimize the speed of BOP for your particular system configuration.

Only a single pair_coeff command is used with the *bop* style which specifies a BOP potential file, with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of BOP elements to atom types

As an example, imagine the CdTe.bop file has BOP values for Cd and Te. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Cd, and the fourth to be Te, you would use the following pair_coeff command:

```
pair_coeff * * CdTe Cd Cd Cd Te
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Cd arguments map LAMMPS atom types 1,2,3 to the Cd element in the BOP file. The final Te argument maps LAMMPS atom type 4 to the Te element in the BOP file.

BOP files in the *potentials* directory of the LAMMPS distribution have a ".bop" suffix. The potentials are in tabulated form containing pre-tabulated pair functions for phi_ij(r_ij), beta_(sigma,ij)(r_ij), and beta_pi,ij)(r_ij).

The parameters/coefficients format for the different kinds of BOP files are given below with variables matching the formulation of Ward (*Ward*) and Zhou (*Zhou*). Each header line containing a ":" is preceded by a blank line.

No angular table file format:

The parameters/coefficients format for the BOP potentials input file containing pre-tabulated functions of g is given below with variables matching the formulation of Ward (*Ward*). This format also assumes the angular functions have the formulation of (*Ward*).

- Line 1: # elements N

The first line is followed by N lines containing the atomic number, mass, and element symbol of each element.

Following the definition of the elements several global variables for the tabulated functions are given.

- Line 1: nr, nBOt (nr is the number of divisions the radius is broken into for function tables and MUST be a factor of 5; nBOt is the number of divisions for the tabulated values of THETA_(S,ij)
- Line 2: delta_1-delta_7 (if all are not used in the particular
- formulation, set unused values to 0.0)

Following this N lines for e_1-e_N containing p_pi.

- Line 3: p_pi (for e_1)
- Line 4: p_pi (for e_2 and continues to e_N)

The next section contains several pair constants for the number of interaction types e_i-e_j, with i=1->N, j=i->N

- Line 1: r_cut (for e_1-e_1 interactions)
- Line 2: c_sigma, a_sigma, c_pi, a_pi
- Line 3: delta_sigma, delta_pi
- Line 4: f_sigma, k_sigma, delta_3 (This delta_3 is similar to that of the previous section but is interaction type dependent)

The next section contains a line for each three body interaction type e_j-e_i-e_k with i=0->N, j=0->N, k=j->N

- Line 1: g_(sigma0), g_(sigma1), g_(sigma2) (These are coefficients for g_(sigma,jik)(THETA_ijk) for e_1-e_1-e_1 interaction. *Ward* contains the full expressions for the constants as functions of b_(sigma,ijk), p_(sigma,ijk), u_(sigma,ijk))

- Line 2: g_(sigma0), g_(sigma1), g_(sigma2) (for e_1-e_1-e_2)

The next section contains a block for each interaction type for the phi_ij(r_ij). Each block has nr entries with 5 entries per line.

- Line 1: phi(r1), phi(r2), phi(r3), phi(r4), phi(r5) (for the e_1-e_1 interaction type)
- Line 2: phi(r6), phi(r7), phi(r8), phi(r9), phi(r10) (this continues until nr)
- …
- Line nr/5_1: phi(r1), phi(r2), phi(r3), phi(r4), phi(r5), (for the e_1-e_1 interaction type)

The next section contains a block for each interaction type for the beta_(sigma,ij)(r_ij). Each block has nr entries with 5 entries per line.

- Line 1: beta_sigma(r1), beta_sigma(r2), beta_sigma(r3), beta_sigma(r4), beta_sigma(r5) (for the e_1-e_1 interaction type)
- Line 2: beta_sigma(r6), beta_sigma(r7), beta_sigma(r8), beta_sigma(r9), beta_sigma(r10) (this continues until nr)
- …
- Line nr/5+1: beta_sigma(r1), beta_sigma(r2), beta_sigma(r3), beta_sigma(r4), beta_sigma(r5) (for the e_1-e_2 interaction type)

The next section contains a block for each interaction type for beta_(pi,ij)(r_ij). Each block has nr entries with 5 entries per line.

- Line 1: beta_pi(r1), beta_pi(r2), beta_pi(r3), beta_pi(r4), beta_pi(r5) (for the e_1-e_1 interaction type)
- Line 2: beta_pi(r6), beta_pi(r7), beta_pi(r8), beta_pi(r9), beta_pi(r10) (this continues until nr)
- …
- Line nr/5+1: beta_pi(r1), beta_pi(r2), beta_pi(r3), beta_pi(r4), beta_pi(r5) (for the e_1-e_2 interaction type)

The next section contains a block for each interaction type for the THETA_(S,ij)((THETA_(sigma,ij))^(1/2), f_(sigma,ij)). Each block has nBOt entries with 5 entries per line.

- Line 1: THETA_(S,ij)(r1), THETA_(S,ij)(r2), THETA_(S,ij)(r3), THETA_(S,ij)(r4), THETA_(S,ij)(r5) (for the e_1-e_2 interaction type)
- Line 2: THETA_(S,ij)(r6), THETA_(S,ij)(r7), THETA_(S,ij)(r8), THETA_(S,ij)(r9), THETA_(S,ij)(r10) (this continues until nBOt)
- …
- Line nBOt/5+1: THETA_(S,ij)(r1), THETA_(S,ij)(r2), THETA_(S,ij)(r3), THETA_(S,ij)(r4), THETA_(S,ij)(r5) (for the e_1-e_2 interaction type)

The next section contains a block of N lines for e_1-e_N

- Line 1: delta^mu (for e_1)
- Line 2: delta^mu (for e_2 and repeats to e_N)

The last section contains more constants for e_i-e_j interactions with i=0->N, j=i->N

- Line 1: (A_ij)^(mu*nu) (for e1-e1)
- Line 2: (A_ij)^(mu*nu) (for e1-e2 and repeats as above)

Angular spline table file format:

The parameters/coefficients format for the BOP potentials input file containing pre-tabulated functions of g is given below with variables matching the formulation of Ward (*Ward*). This format also assumes the angular functions have the formulation of (*Zhou*).

- Line 1: # elements N

The first line is followed by N lines containing the atomic number, mass, and element symbol of each element.

Following the definition of the elements several global variables for the tabulated functions are given.

- Line 1: nr, ntheta, nBOt (nr is the number of divisions the radius is broken into for function tables and MUST be a factor of 5; ntheta is the power of the power of the spline used to fit the angular function; nBOt is the number of divisions for the tabulated values of THETA_(S,ij)
- Line 2: delta_1-delta_7 (if all are not used in the particular
- formulation, set unused values to 0.0)

Following this N lines for e_1-e_N containing p_pi.

- Line 3: p_pi (for e_1)
- Line 4: p_pi (for e_2 and continues to e_N)

The next section contains several pair constants for the number of interaction types e_i-e_j, with i=1->N, j=i->N

- Line 1: r_cut (for e_1-e_1 interactions)
- Line 2: c_sigma, a_sigma, c_pi, a_pi
- Line 3: delta_sigma, delta_pi
- Line 4: f_sigma, k_sigma, delta_3 (This delta_3 is similar to that of the previous section but is interaction type dependent)

The next section contains a line for each three body interaction type e_j-e_i-e_k with i=0->N, j=0->N, k=j->N

- Line 1: g0, g1, g2… (These are coefficients for the angular spline of the g_(sigma,jik)(THETA_ijk) for e_1-e_1-e_1 interaction. The function can contain up to 10 term thus 10 constants. The first line can contain up to five constants. If the spline has more than five terms the second line will contain the remaining constants The following lines will then contain the constants for the remaining g0, g1, g2… (for e_1-e_1-e_2) and the other three body interactions

The rest of the table has the same structure as the previous section (see above).

Angular no-spline table file format:

The parameters/coefficients format for the BOP potentials input file containing pre-tabulated functions of g is given below with variables matching the formulation of Ward (*Ward*). This format also assumes the angular functions have the formulation of (*Zhou*).

- Line 1: # elements N

The first two lines are followed by N lines containing the atomic number, mass, and element symbol of each element.

Following the definition of the elements several global variables for the tabulated functions are given.

- Line 1: nr, ntheta, nBOt (nr is the number of divisions the radius is broken into for function tables and MUST be a factor of 5; ntheta is the number of divisions for the tabulated values of the g angular function; nBOt is the number of divisions for the tabulated values of THETA_(S,ij)
- Line 2: delta_1-delta_7 (if all are not used in the particular
- formulation, set unused values to 0.0)

Following this N lines for e_1-e_N containing p_pi.

- Line 3: p_pi (for e_1)
- Line 4: p_pi (for e_2 and continues to e_N)

The next section contains several pair constants for the number of interaction types e_i-e_j, with i=1->N, j=i->N

- Line 1: r_cut (for e_1-e_1 interactions)
- Line 2: c_sigma, a_sigma, c_pi, a_pi
- Line 3: delta_sigma, delta_pi
- Line 4: f_sigma, k_sigma, delta_3 (This delta_3 is similar to that of the previous section but is interaction type dependent)

The next section contains a line for each three body interaction type e_j-e_i-e_k with i=0->N, j=0->N, k=j->N

- Line 1: g(theta1), g(theta2), g(theta3), g(theta4), g(theta5) (for the e_1-e_1-e_1 interaction type)
- Line 2: g(theta6), g(theta7), g(theta8), g(theta9), g(theta10) (this continues until ntheta)
- ...
- Line ntheta/5+1: g(theta1), g(theta2), g(theta3), g(theta4), g(theta5), (for the e_1-e_1-e_2 interaction type)

The rest of the table has the same structure as the previous section (see above).

4.15.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.15.5 Restrictions

These pair styles are part of the MANYBODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair potentials require the *newton* setting to be "on" for pair interactions.

Pair style bop is not compatible with being used as a sub-style with doc:*hybrid pair styles <pair_hybrid>*. Pair style bop is also not compatible with *multi-cutoff neighbor lists* or *multi-cutoff communitcation*.

The .bop.table potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the BOP potential with any LAMMPS units, but you would need to create your own BOP potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.15.6 Related commands

pair_coeff

4.15.7 Default

non-tabulated potential file, a_0 is non-zero.

(Pettifor_1) D.G. Pettifor and I.I. Oleinik, Phys. Rev. B, 59, 8487 (1999).

(Pettifor_2) D.G. Pettifor and I.I. Oleinik, Phys. Rev. Lett., 84, 4124 (2000).

(Pettifor_3) D.G. Pettifor and I.I. Oleinik, Phys. Rev. B, 65, 172103 (2002).

(Murdick) D.A. Murdick, X.W. Zhou, H.N.G. Wadley, D. Nguyen-Manh, R. Drautz, and D.G. Pettifor, Phys. Rev. B, 73, 45206 (2006).

(Ward) D.K. Ward, X.W. Zhou, B.M. Wong, F.P. Doty, and J.A. Zimmerman, Phys. Rev. B, 85,115206 (2012).

(Zhou) X.W. Zhou, D.K. Ward, M. Foster (TBP).

4.16 pair_style born command

Accelerator Variants: *born/omp*, *born/gpu*

4.17 pair_style born/coul/long command

Accelerator Variants: *born/coul/long/gpu*, *born/coul/long/omp*

4.18 pair_style born/coul/msm command

Accelerator Variants: *born/coul/msm/omp*

4.19 pair_style born/coul/wolf command

Accelerator Variants: *born/coul/wolf/gpu*, *born/coul/wolf/omp*

4.20 pair_style born/coul/dsf command

4.20.1 Syntax

```
pair_style style args
```

- style = *born* or *born/coul/long* or *born/coul/msm* or *born/coul/wolf*
- args = list of arguments for a particular style

born args = cutoff
  cutoff = global cutoff for non-Coulombic interactions (distance units)
born/coul/long args = cutoff (cutoff2)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
born/coul/msm args = cutoff (cutoff2)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
born/coul/wolf args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
born/coul/dsf args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (distance units)

4.20.2 Examples

```
pair_style born 10.0
pair_coeff * * 6.08 0.317 2.340 24.18 11.51
pair_coeff 1 1 6.08 0.317 2.340 24.18 11.51

pair_style born/coul/long 10.0
pair_style born/coul/long 10.0 8.
pair_coeff * * 6.08 0.317 2.340 24.18 11.51
pair_coeff 1 1 6.08 0.317 2.340 24.18 11.51

pair_style born/coul/msm 10.0
pair_style born/coul/msm 10.0 8.0
pair_coeff * * 6.08 0.317 2.340 24.18 11.51
pair_coeff 1 1 6.08 0.317 2.340 24.18 11.51

pair_style born/coul/wolf 0.25 10.0
pair_style born/coul/wolf 0.25 10.0 9.0
pair_coeff * * 6.08 0.317 2.340 24.18 11.51
pair_coeff 1 1 6.08 0.317 2.340 24.18 11.51

pair_style born/coul/dsf 0.1 10.0 12.0
pair_coeff * *   0.0 1.00 0.00 0.00 0.00
pair_coeff 1 1 480.0 0.25 0.00 1.05 0.50
```

4.20.3 Description

The *born* style computes the Born-Mayer-Huggins or Tosi/Fumi potential described in *(Fumi and Tosi)*, given by

$$E = A \exp\left(\frac{\sigma - r}{\rho}\right) - \frac{C}{r^6} + \frac{D}{r^8} \qquad r < r_c$$

where σ is an interaction-dependent length parameter, ρ is an ionic-pair dependent length parameter, and r_c is the cutoff.

The styles with *coul/long* or *coul/msm* add a Coulombic term as described for the *lj/cut* pair styles. An additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* of *msm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

If one cutoff is specified for the *born/coul/long* and *born/coul/msm* style, it is used for both the A,C,D and Coulombic terms. If two cutoffs are specified, the first is used as the cutoff for the A,C,D terms, and the second is the cutoff for the Coulombic term.

The *born/coul/wolf* style adds a Coulombic term as described for the Wolf potential in the *coul/wolf* pair style.

The *born/coul/dsf* style computes the Coulomb contribution with the damped shifted force model as in the *coul/dsf* style.

Note that these potentials are related to the *Buckingham potential*.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- A (energy units)
- ρ (distance units)
- σ (distance units)
- C (energy units * distance units^6)
- D (energy units * distance units^8)
- cutoff (distance units)

The second coefficient, rho, must be greater than zero.

The last coefficient is optional. If not specified, the global A,C,D cutoff specified in the pair_style command is used.

For *born/coul/long*, *born/coul/wolf* and *born/coul/dsf* no Coulombic cutoff can be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.20.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

These styles support the *pair_modify* shift option for the energy of the exp(), 1/r^6, and 1/r^8 portion of the pair interaction.

The *born/coul/long* pair style supports the *pair_modify* table option to tabulate the short-range portion of the long-range Coulombic interaction.

These styles support the pair_modify tail option for adding long-range tail corrections to energy and pressure.

Thess styles writes thei information to binary *restart* files, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.20.5 Restrictions

The *born/coul/long* style is part of the KSPACE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The *born/coul/dsf* and *born/coul/wolf* pair styles are part of the EXTRA-PAIR package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.20.6 Related commands

pair_coeff, *pair_style buck*

4.20.7 Default

none

Fumi and Tosi, J Phys Chem Solids, 25, 31 (1964), Fumi and Tosi, J Phys Chem Solids, 25, 45 (1964).

4.21 pair_style born/gauss command

4.21.1 Syntax

```
pair_style born/gauss cutoff
```

- born/gauss = name of the pair style
- cutoff = global cutoff (distance units)

4.21.2 Examples

```
pair_style born/gauss 10.0
pair_coeff 1 1 8.2464e13 12.48 0.042644277 0.44 3.56
```

4.21.3 Description

Added in version 28Mar2023.

Pair style *born/gauss* computes pairwise interactions from a combination of a Born-Mayer repulsive term and a Gaussian attractive term according to *(Bomont)*:

$$E = A_0 \exp\left(-\alpha r\right) - A_1 \exp\left[-\beta \left(r - r_0\right)^2\right] \qquad r < r_c$$

r_c is the cutoff.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A_0 (energy units)
- α (1/distance units)
- A_1 (energy units)
- β (1/(distance units)^2)
- r_0 (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

4.21.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table options are not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.21.5 Restrictions

This pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

4.21.6 Related commands

pair_coeff, *pair_style born*

4.21.7 Default

none

(Bomont) Bomont, Bretonnet, J. Chem. Phys. 124, 054504 (2006)

4.22 pair_style bpm/spring command

4.22.1 Syntax

```
pair_style bpm/spring
```

4.22.2 Examples

```
pair_style bpm/spring
pair_coeff * * 1.0 1.0 1.0
pair_coeff 1 1 1.0 1.0 1.0
```

4.22.3 Description

Added in version 4May2022.

Style *bpm/spring* computes pairwise forces with the formula

$$F = k(r - r_c)$$

where k is a stiffness and r_c is the cutoff length. An additional damping force is also applied to interacting particles. The force is proportional to the difference in the normal velocity of particles

$$F_D = -\gamma w(\hat{r} \bullet \vec{v})$$

where γ is the damping strength, $\hat{r}$ is the radial normal vector, $\vec{v}$ is the velocity difference between the two particles, and w is a smoothing factor. This smoothing factor is constructed such that damping forces go to zero as particles come out of contact to avoid discontinuities. It is given by

$$w = 1.0 - \left(\frac{r}{r_c}\right)^8.$$

This pair style is designed for use in a spring-based bonded particle model. It mirrors the construction of the *bpm/spring* bond style.

This pair interaction is always applied to pairs of non-bonded particles that are within the interaction distance. For pairs of bonded particles that are within the interaction distance, there is the option to either include this pair interaction and overlay the pair force over the bond force or to exclude this pair interaction such that the two particles only interact via the bond force. See discussion of the *overlay/pair* option for BPM bond styles and the *special_bonds* command in the *how to* page on BPMs for more details.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- k (force/distance units)
- r_c (distance units)
- γ (force/velocity units)

4.22.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A coefficient and cutoff distance for this pair style can be mixed. A is always mixed via a *geometric* rule. The cutoff is mixed according to the pair_modify mix value. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option, since the pair interaction goes to 0.0 at the cutoff.

The *pair_modify* table and tail options are not relevant for this pair style.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.22.5 Restrictions

This pair style is part of the BPM package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.22.6 Related commands

pair_coeff, *bond bpm/spring*

4.22.7 Default

none

4.23 pair_style brownian command

Accelerator Variants: *brownian/omp*

4.24 pair_style brownian/poly command

Accelerator Variants: *brownian/poly/omp*

4.24.1 Syntax

```
pair_style style mu flaglog flagfld cutinner cutoff t_target seed flagHI flagVF
```

- style = *brownian* or *brownian/poly*
- mu = dynamic viscosity (dynamic viscosity units)
- flaglog = 0/1 log terms in the lubrication approximation on/off
- flagfld = 0/1 to include/exclude Fast Lubrication Dynamics effects
- cutinner = inner cutoff distance (distance units)
- cutoff = outer cutoff for interactions (distance units)
- t_target = target temp of the system (temperature units)
- seed = seed for the random number generator (positive integer)
- flagHI (optional) = 0/1 to include/exclude 1/r hydrodynamic interactions
- flagVF (optional) = 0/1 to include/exclude volume fraction corrections in the long-range isotropic terms

4.24.2 Examples

```
pair_style brownian 1.5 1 1 2.01 2.5 2.0 5878567 # (assuming radius = 1)
pair_coeff 1 1 2.05 2.8
pair_coeff * *
```

4.24.3 Description

Styles *brownian* and *brownian/poly* compute Brownian forces and torques on finite-size spherical particles. The former requires monodisperse spherical particles; the latter allows for polydisperse spherical particles.

These pair styles are designed to be used with either the *pair_style lubricate* or *pair_style lubricateU* commands to provide thermostatting when dissipative lubrication forces are acting. Thus the parameters *mu*, *flaglog*, *flagfld*, *cutinner*, and *cutoff* should be specified consistent with the settings in the lubrication pair styles. For details, refer to either of the lubrication pair styles.

The *t_target* setting is used to specify the target temperature of the system. The random number *seed* is used to generate random numbers for the thermostatting procedure.

The *flagHI* and *flagVF* settings are optional. Neither should be used, or both must be defined.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutinner (distance units)
- cutoff (distance units)

The two coefficients are optional. If neither is specified, the two cutoffs specified in the pair_style command are used. Otherwise both must be specified.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.24.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the two cutoff distances for these pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

These pair styles do not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for these pair styles.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.24.5 Restrictions

These styles are part of the COLLOID package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Only spherical monodisperse particles are allowed for pair_style brownian.

Only spherical particles are allowed for pair_style brownian/poly.

These pair styles are only compatible with the following wall fixes: doc:*fix wall/lj93, fix wall/lj126, fix wall/lj1043, fix wall/colloid, fix wall/harmonic, fix wall/lepton, fix wall/morse, fix wall/table <fix_wall>*.

4.24.6 Related commands

pair_coeff , *pair_style lubricate*, *pair_style lubricateU*

4.24.7 Default

The default settings for the optional args are flagHI = 1 and flagVF = 1.

4.25 pair_style buck command

Accelerator Variants: *buck/gpu*, *buck/intel*, *buck/kk*, *buck/omp*

4.26 pair_style buck/coul/cut command

Accelerator Variants: *buck/coul/cut/gpu*, *buck/coul/cut/intel*, *buck/coul/cut/kk*, *buck/coul/cut/omp*

4.27 pair_style buck/coul/long command

Accelerator Variants: *buck/coul/long/gpu*, *buck/coul/long/intel*, *buck/coul/long/kk*, *buck/coul/long/omp*

4.28 pair_style buck/coul/msm command

Accelerator Variants: *buck/coul/msm/omp*

4.28.1 Syntax

```
pair_style style args
```

- style = *buck* or *buck/coul/cut* or *buck/coul/long* or *buck/coul/msm*
- args = list of arguments for a particular style

buck args = cutoff
  cutoff = global cutoff for Buckingham interactions (distance units)
buck/coul/cut args = cutoff (cutoff2)
  cutoff = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
buck/coul/long args = cutoff (cutoff2)
  cutoff = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
buck/coul/msm args = cutoff (cutoff2)
  cutoff = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.28.2 Examples

```
pair_style buck 2.5
pair_coeff * * 100.0 1.5 200.0
pair_coeff * * 100.0 1.5 200.0 3.0

pair_style buck/coul/cut 10.0
pair_style buck/coul/cut 10.0 8.0
pair_coeff * * 100.0 1.5 200.0
pair_coeff 1 1 100.0 1.5 200.0 9.0
pair_coeff 1 1 100.0 1.5 200.0 9.0 8.0

pair_style buck/coul/long 10.0
pair_style buck/coul/long 10.0 8.0
pair_coeff * * 100.0 1.5 200.0
pair_coeff 1 1 100.0 1.5 200.0 9.0

pair_style buck/coul/msm 10.0
pair_style buck/coul/msm 10.0 8.0
pair_coeff * * 100.0 1.5 200.0
pair_coeff 1 1 100.0 1.5 200.0 9.0
```

4.28.3 Description

The *buck* style computes a Buckingham potential (exp/6 instead of Lennard-Jones 12/6) given by

$$E = Ae^{-r/\rho} - \frac{C}{r^6} \qquad r < r_c$$

where ρ is an ionic-pair dependent length parameter, and r_c is the cutoff on both terms.

The styles with *coul/cut* or *coul/long* or *coul/msm* add a Coulombic term as described for the *lj/cut* pair styles. For *buck/coul/long* and *buc/coul/msm*, an additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* or *msm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

If one cutoff is specified for the *born/coul/cut* and *born/coul/long* and *born/coul/msm* styles, it is used for both the A,C and Coulombic terms. If two cutoffs are specified, the first is used as the cutoff for the A,C terms, and the second is the cutoff for the Coulombic term.

Note that these potentials are related to the *Born-Mayer-Huggins potential*.

Note

For all these pair styles, the terms with A and C are always cutoff. The additional Coulombic term can be cutoff or long-range (no cutoff) depending on whether the style name includes coul/cut or coul/long or coul/msm. If you wish the C/r^6 term to be long-range (no cutoff), then see the *pair_style buck/long/coul/long* command.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (energy units)
- ρ (distance units)
- C (energy-distance^6 units)
- cutoff (distance units)
- cutoff2 (distance units)

The second coefficient, ρ, must be greater than zero. The coefficients A, ρ, and C can be written as analytical expressions of ε and σ, in analogy to the Lennard-Jones potential *(Khrapak)*.

The latter 2 coefficients are optional. If not specified, the global A,C and Coulombic cutoffs are used. If only one cutoff is specified, it is used as the cutoff for both A,C and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the A,C and Coulombic cutoffs for this type pair. You cannot specify 2 cutoffs for style *buck*, since it has no Coulombic terms. For *buck/coul/long* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.28.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

These styles support the *pair_modify* shift option for the energy of the exp() and 1/r^6 portion of the pair interaction.

The *buck/coul/long* pair style supports the *pair_modify* table option to tabulate the short-range portion of the long-range Coulombic interaction.

These styles support the pair_modify tail option for adding long-range tail corrections to energy and pressure for the A,C terms in the pair interaction.

These styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.28.5 Restrictions

The *buck/coul/long* style is part of the KSPACE package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.28.6 Related commands

pair_coeff, *pair_style born*

4.28.7 Default

none

(Khrapak) Khrapak, Chaudhuri, and Morfill, J Chem Phys, 134, 054120 (2011).

4.29 pair_style buck6d/coul/gauss/dsf command

4.30 pair_style buck6d/coul/gauss/long command

4.30.1 Syntax

```
pair_style style args
```

- style = *buck6d/coul/gauss/dsf* or *buck6d/coul/gauss/long*
- args = list of arguments for a particular style

```
buck6d/coul/gauss/dsf args = smooth cutoff (cutoff2)
  smooth  = smoothing onset within Buckingham cutoff (ratio)
  cutoff  = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
buck6d/coul/gauss/long args = smooth smooth2 cutoff (cutoff2)
  smooth   = smoothing onset within Buckingham cutoff (ratio)
  smooth2  = smoothing onset within Coulombic cutoff (ratio)
  cutoff   = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2  = global cutoff for Coulombic (optional) (distance units)
```

4.30.2 Examples

```
pair_style buck6d/coul/gauss/dsf    0.9000    12.0000
pair_coeff 1  1  1030.  3.061  457.179  4.521  0.608

pair_style buck6d/coul/gauss/long   0.9000  1.0000  12.0000
pair_coeff 1  1  1030.  3.061  457.179  4.521  0.608
```

4.30.3 Description

The *buck6d/coul/gauss* styles evaluate vdW and Coulomb interactions following the MOF-FF force field after *(Schmid)*. The vdW term of the *buck6d* styles computes a dispersion damped Buckingham potential:

$$E = Ae^{-\kappa r} - \frac{C}{r^6} \cdot \frac{1}{1 + Dr^{14}} \qquad r < r_c$$

where A and C are a force constant, κ is an ionic-pair dependent reciprocal length parameter, D is a dispersion correction parameter, and the cutoff r_c truncates the interaction distance. The first term in the potential corresponds to the Buckingham repulsion term and the second term to the dispersion attraction with a damping correction analog to the Grimme correction used in DFT. The latter corrects for artifacts occurring at short distances which become an issue for soft vdW potentials.

The *buck6d* styles include a smoothing function which is invoked according to the global smoothing parameter within the specified cutoff. Hereby a parameter of i.e. 0.9 invokes the smoothing within 90% of the cutoff. No smoothing is applied at a value of 1.0. For the *gauss/dsf* style this smoothing is only applicable for the dispersion damped Buckingham potential. For the *gauss/long* styles the smoothing function can also be invoked for the real space coulomb interactions which enforce continuous energies and forces at the cutoff.

Both styles *buck6d/coul/gauss/dsf* and *buck6d/coul/gauss/long* evaluate a Coulomb potential using spherical Gaussian type charge distributions which effectively dampen electrostatic interactions for high charges at close distances. The electrostatic potential is thus evaluated as:

$$E = \frac{C_{q_i q_j}}{\varepsilon r_{ij}} \operatorname{erf}\left(\alpha_{ij} r_{ij}\right) \qquad r < r_c$$

where C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, epsilon is the dielectric constant which can be set by the *dielectric* command, α is the ion pair dependent damping parameter and erf() is the error-function. The cutoff r_c truncates the interaction distance.

The style *buck6d/coul/gauss/dsf* computes the Coulomb interaction via the damped shifted force model described in *(Fennell)* approximating an Ewald sum similar to the *pair coul/dsf* styles. In *buck6d/coul/gauss/long* an additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* options. The Coulombic cutoff in this case separates the real and reciprocal space evaluation of the Ewald sum.

If one cutoff is specified it is used for both the vdW and Coulomb terms. If two cutoffs are specified, the first is used as the cutoff for the vdW terms, and the second is the cutoff for the Coulombic term.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (energy units)
- ρ (distance^-1 units)
- C (energy-distance^6 units)
- D (distance^14 units)

- α (distance^-1 units)
- cutoff (distance units)

The second coefficient, ρ, must be greater than zero. The latter coefficient is optional. If not specified, the global vdW cutoff is used.

4.30.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

These styles do not support the *pair_modify* shift option for the energy. Instead the smoothing function should be applied by setting the global smoothing parameter to a value < 1.0.

These styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

4.30.5 Restrictions

These styles are part of the MOFFF package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.30.6 Related commands

pair_coeff

4.30.7 Default

none

(Schmid) S. Bureekaew, S. Amirjalayer, M. Tafipolsky, C. Spickermann, T.K. Roy and R. Schmid, Phys. Status Solidi B, 6, 1128 (2013).

(Fennell) C. J. Fennell, J. D. Gezelter, J Chem Phys, 124, 234104 (2006).

4.31 pair_style buck/long/coul/long command

Accelerator Variants: *buck/long/coul/long/omp*

4.31.1 Syntax

```
pair_style buck/long/coul/long flag_buck flag_coul cutoff (cutoff2)
```

- flag_buck = *long* or *cut*

  long = use Kspace long-range summation for the dispersion term 1/r^6
  cut = use a cutoff

- flag_coul = *long* or *off*

long = use Kspace long-range summation for the Coulombic term 1/r
off = omit the Coulombic term

- cutoff = global cutoff for Buckingham (and Coulombic if only 1 cutoff) (distance units)
- cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.31.2 Examples

```
pair_style buck/long/coul/long cut off 2.5
pair_style buck/long/coul/long cut long 2.5 4.0
pair_style buck/long/coul/long long long 4.0
pair_coeff * * 1 1
pair_coeff 1 1 1 3 4
```

4.31.3 Description

The *buck/long/coul/long* style computes a Buckingham potential (exp/6 instead of Lennard-Jones 12/6) and Coulombic potential, given by

$$E = Ae^{-r/\rho} - \frac{C}{r^6} \qquad r < r_c$$

$$E = \frac{Cq_iq_j}{\varepsilon r} \qquad r < r_c$$

r_c is the cutoff. If one cutoff is specified in the pair_style command, it is used for both the Buckingham and Coulombic terms. If two cutoffs are specified, they are used as cutoffs for the Buckingham and Coulombic terms respectively.

The purpose of this pair style is to capture long-range interactions resulting from both attractive 1/r^6 Buckingham and Coulombic 1/r interactions. This is done by use of the *flag_buck* and *flag_coul* settings. The *Ismail* paper has more details on when it is appropriate to include long-range 1/r^6 interactions, using this potential.

If *flag_buck* is set to *long*, no cutoff is used on the Buckingham 1/r^6 dispersion term. The long-range portion can be calculated by using the *kspace_style ewald/disp or pppm/disp* commands. The specified Buckingham cutoff then determines which portion of the Buckingham interactions are computed directly by the pair potential versus which part is computed in reciprocal space via the Kspace style. If *flag_buck* is set to *cut*, the Buckingham interactions are simply cutoff, as with *pair_style buck*.

If *flag_coul* is set to *long*, no cutoff is used on the Coulombic interactions. The long-range portion can calculated by using any of several *kspace_style* command options such as *pppm* or *ewald*. Note that if *flag_buck* is also set to long, then the *ewald/disp* or *pppm/disp* Kspace style needs to be used to perform the long-range calculations for both the Buckingham and Coulombic interactions. If *flag_coul* is set to *off*, Coulombic interactions are not computed.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (energy units)
- rho (distance units)
- C (energy-distance^6 units)
- cutoff (distance units)
- cutoff2 (distance units)

The second coefficient, rho, must be greater than zero.

The latter 2 coefficients are optional. If not specified, the global Buckingham and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both Buckingham and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the Buckingham and Coulombic cutoffs for this type pair. Note that if you are using *flag_buck* set to *long*, you cannot specify a Buckingham cutoff for an atom type pair, since only one global Buckingham cutoff is allowed. Similarly, if you are using *flag_coul* set to *long*, you cannot specify a Coulombic cutoff for an atom type pair, since only one global Coulombic cutoff is allowed.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.31.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This pair style supports the *pair_modify* shift option for the energy of the exp() and 1/r^6 portion of the pair interaction, assuming *flag_buck* is *cut*.

This pair style does not support the *pair_modify* shift option for the energy of the Buckingham portion of the pair interaction.

This pair style supports the *pair_modify* table and table/disp options since they can tabulate the short-range portion of the long-range Coulombic and dispersion interactions.

This pair style write its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style supports the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. See the *run_style* command for details.

4.31.5 Restrictions

This style is part of the KSPACE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.31.6 Related commands

pair_coeff

4.31.7 Default

none

(Ismail) Ismail, Tsige, In 't Veld, Grest, Molecular Physics (accepted) (2007).

4.32 pair_style lj/charmm/coul/charmm command

Accelerator Variants: *lj/charmm/coul/charmm/gpu*, *lj/charmm/coul/charmm/intel*, *lj/charmm/coul/charmm/kk*, *lj/charmm/coul/charmm/omp*

4.33 pair_style lj/charmm/coul/charmm/implicit command

Accelerator Variants: *lj/charmm/coul/charmm/implicit/kk*, *lj/charmm/coul/charmm/implicit/omp*

4.34 pair_style lj/charmm/coul/long command

Accelerator Variants: *lj/charmm/coul/long/gpu*, *lj/charmm/coul/long/intel*, *lj/charmm/coul/long/kk*, *lj/charmm/coul/long/opt*, *lj/charmm/coul/long/omp*

4.35 pair_style lj/charmm/coul/msm command

Accelerator Variants: *lj/charmm/coul/msm/omp*

4.36 pair_style lj/charmmfsw/coul/charmmfsh command

4.37 pair_style lj/charmmfsw/coul/long command

Accelerator Variants: *lj/charmmfsw/coul/long/kk*

4.37.1 Syntax

```
pair_style style args
```

- style = *lj/charmm/coul/charmm* or *lj/charmm/coul/charmm/implicit* or *lj/charmm/coul/long* or *lj/charmm/coul/msm* or *lj/charmmfsw/coul/charmmfsh* or *lj/charmmfsw/coul/long*
- args = list of arguments for a particular style

lj/charmm/coul/charmm args = inner outer (inner2) (outer2)
  inner, outer = global switching cutoffs for Lennard Jones (and Coulombic if only 2 args)
  inner2, outer2 = global switching cutoffs for Coulombic (optional)
lj/charmm/coul/charmm/implicit args = inner outer (inner2) (outer2)
  inner, outer = global switching cutoffs for LJ (and Coulombic if only 2 args)
  inner2, outer2 = global switching cutoffs for Coulombic (optional)
lj/charmm/coul/long args = inner outer (cutoff)
  inner, outer = global switching cutoffs for LJ (and Coulombic if only 2 args)
  cutoff = global cutoff for Coulombic (optional, outer is Coulombic cutoff if only 2 args)
lj/charmm/coul/msm args = inner outer (cutoff)
  inner, outer = global switching cutoffs for LJ (and Coulombic if only 2 args)
  cutoff = global cutoff for Coulombic (optional, outer is Coulombic cutoff if only 2 args)
lj/charmmfsw/coul/charmmfsh args = inner outer (cutoff)
  inner, outer = global cutoffs for LJ (and Coulombic if only 2 args)
  cutoff = global cutoff for Coulombic (optional, outer is Coulombic cutoff if only 2 args)
lj/charmmfsw/coul/long args = inner outer (cutoff)
  inner, outer = global cutoffs for LJ (and Coulombic if only 2 args)
  cutoff = global cutoff for Coulombic (optional, outer is Coulombic cutoff if only 2 args)

4.37.2 Examples

```
pair_style lj/charmm/coul/charmm 8.0 10.0
pair_style lj/charmm/coul/charmm 8.0 10.0 7.0 9.0
pair_style lj/charmmfsw/coul/charmmfsh 10.0 12.0
pair_style lj/charmmfsw/coul/charmmfsh 10.0 12.0 9.0
pair_coeff * * 100.0 2.0
pair_coeff 1 1 100.0 2.0 150.0 3.5

pair_style lj/charmm/coul/charmm/implicit 8.0 10.0
pair_style lj/charmm/coul/charmm/implicit 8.0 10.0 7.0 9.0
pair_coeff * * 100.0 2.0
pair_coeff 1 1 100.0 2.0 150.0 3.5

pair_style lj/charmm/coul/long 8.0 10.0
pair_style lj/charmm/coul/long 8.0 10.0 9.0
pair_style lj/charmmfsw/coul/long 8.0 10.0
pair_style lj/charmmfsw/coul/long 8.0 10.0 9.0
pair_coeff * * 100.0 2.0
pair_coeff 1 1 100.0 2.0 150.0 3.5

pair_style lj/charmm/coul/msm 8.0 10.0
pair_style lj/charmm/coul/msm 8.0 10.0 9.0
pair_coeff * * 100.0 2.0
pair_coeff 1 1 100.0 2.0 150.0 3.5
```

4.37.3 Description

These pair styles compute Lennard Jones (LJ) and Coulombic interactions with additional switching or shifting functions that ramp the energy and/or force smoothly to zero between an inner and outer cutoff. They implement the widely used CHARMM force field, see *Howto discussion on biomolecular force fields* for details.

The styles with *charmm* (not *charmmfsw* or *charmmfsh*) in their name are the older, original LAMMPS implementations. They compute the LJ and Coulombic interactions with an energy switching function which ramps the energy smoothly to zero between the inner and outer cutoff. This can cause irregularities in pairwise forces (due to the discontinuous second derivative of energy at the boundaries of the switching region), which in some cases can result in detectable artifacts in an MD simulation.

The newer styles with *charmmfsw* or *charmmfsh* in their name replace the energy switching with force switching (fsw) and force shifting (fsh) functions, for LJ and Coulombic interactions respectively.

Note

The newer *charmmfsw* or *charmmfsh* styles were released in March 2017. We recommend they be used instead of the older *charmm* styles. This includes the newer *dihedral_style charmmfsw* command. Eventually code from the new styles will propagate into the related pair styles (e.g. implicit, accelerator, free energy variants).

Note

The newest CHARMM pair styles reset the Coulombic energy conversion factor used internally in the code, from the LAMMPS value to the CHARMM value, as if it were effectively a parameter of the force field. This is because the CHARMM code uses a slightly different value for the this conversion factor in *real units* (kcal/mol), namely CHARMM = 332.0716, LAMMPS = 332.06371. This is to enable more precise agreement by LAMMPS with the CHARMM force field energies and forces, when using one of these two CHARMM pair styles.

When using the *lj/charmm/coul/charmm styles*, both the LJ and Coulombic terms require an inner and outer cutoff. They can be the same for both formulas or different depending on whether 2 or 4 arguments are used in the pair_style command. For the *lj/charmmfsw/coul/charmmfsh* style, the LJ term requires both an inner and outer cutoff, while the Coulombic term requires only one cutoff. If the Coulombic cutoff is not specified (2 instead of 3 arguments), the LJ outer cutoff is used for the Coulombic cutoff. In all cases where an inner and outer cutoff are specified, the inner cutoff distance must be less than the outer cutoff. It is typical to make the difference between the inner and outer cutoffs about 2.0 Angstroms.

Style *lj/charmm/coul/charmm/implicit* computes the same formulas as style *lj/charmm/coul/charmm* except that an additional 1/r term is included in the Coulombic formula. The Coulombic energy thus varies as 1/r^2. This is effectively a distance-dependent dielectric term which is a simple model for an implicit solvent with additional screening. It is designed for use in a simulation of an unsolvated biomolecule (no explicit water molecules).

Styles *lj/charmm/coul/long* and *lj/charmm/coul/msm* compute the same formulas as style *lj/charmm/coul/charmm* and style *lj/charmmfsw/coul/long* computes the same formulas as style *lj/charmmfsw/coul/charmmfsh*, except that an additional damping factor is applied to the Coulombic term, so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* or *msm* option. Only one Coulombic cutoff is specified for these styles; if only 2 arguments are used in the pair_style command, then the outer LJ cutoff is used as the single Coulombic cutoff. The Coulombic cutoff specified for these styles means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- ε_{14} (energy units)
- σ_{14} (distance units)

Note that σ is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum at $2^{1/6}\sigma$.

The latter 2 coefficients are optional. If they are specified, they are used in the LJ formula between two atoms of these types which are also first and fourth atoms in any dihedral. No cutoffs are specified because the CHARMM force field does not allow varying cutoffs for individual atom pairs; all pairs use the global cutoff(s) specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.37.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon, sigma, epsilon_14, and sigma_14 coefficients for all of the lj/charmm pair styles can be mixed. The default mix value is *arithmetic* to coincide with the usual settings for the CHARMM force field. See the "pair_modify" command for details.

None of the *lj/charmm* or *lj/charmmfsw* pair styles support the *pair_modify* shift option, since the Lennard-Jones portion of the pair interaction is smoothed to 0.0 at the cutoff.

The *lj/charmm/coul/long* and *lj/charmmfsw/coul/long* styles support the *pair_modify* table option since they can tabulate the short-range portion of the long-range Coulombic interaction.

None of the *lj/charmm* or *lj/charmmfsw* pair styles support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since the Lennard-Jones portion of the pair interaction is smoothed to 0.0 at the cutoff.

All of the *lj/charmm* and *lj/charmmfsw* pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The *lj/charmm/coul/long* and *lj/charmmfsw/coul/long* pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. The other styles only support the *pair* keyword of run_style respa. See the *run_style* command for details.

4.37.5 Restrictions

All the styles with *coul/charmm* or *coul/charmmfsh* styles are part of the MOLECULE package. All the styles with *coul/long* style are part of the KSPACE package. They are only enabled if LAMMPS was built with those packages. See the *Build package* doc page for more info.

4.37.6 Related commands

pair_coeff, *angle_style charmm*, *dihedral_style charmm*, *dihedral_style charmmfsw*, *fix cmap*

4.37.7 Default

none

(Brooks) Brooks, et al, J Comput Chem, 30, 1545 (2009).

(MacKerell) MacKerell, Bashford, Bellott, Dunbrack, Evanseck, Field, Fischer, Gao, Guo, Ha, et al, J Phys Chem, 102, 3586 (1998).

(Steinbach) Steinbach, Brooks, J Comput Chem, 15, 667 (1994).

4.38 pair_style lj/class2 command

Accelerator Variants: *lj/class2/gpu*, *lj/class2/kk*, *lj/class2/omp*

4.39 pair_style lj/class2/coul/cut command

Accelerator Variants: *lj/class2/coul/cut/kk*, *lj/class2/coul/cut/omp*

4.40 pair_style lj/class2/coul/long command

Accelerator Variants: *lj/class2/coul/long/gpu*, *lj/class2/coul/long/kk*, *lj/class2/coul/long/omp*

4.40.1 Syntax

```
pair_style style args
```

- style = *lj/class2* or *lj/class2/coul/cut* or *lj/class2/coul/long*
- args = list of arguments for a particular style

```
lj/class2 args = cutoff
  cutoff = global cutoff for class 2 interactions (distance units)
lj/class2/coul/cut args = cutoff (cutoff2)
  cutoff = global cutoff for class 2 (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/class2/coul/long args = cutoff (cutoff2)
```

cutoff = global cutoff for class 2 (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.40.2 Examples

```
pair_style lj/class2 10.0
pair_coeff * * 100.0 2.5
pair_coeff 1 2* 100.0 2.5 9.0

pair_style lj/class2/coul/cut 10.0
pair_style lj/class2/coul/cut 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0
pair_coeff 1 1 100.0 3.5 9.0 9.0

pair_style lj/class2/coul/long 10.0
pair_style lj/class2/coul/long 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0
```

4.40.3 Description

The *lj/class2* styles compute a 6/9 Lennard-Jones potential given by

$$E = \varepsilon \left[2 \left(\frac{\sigma}{r} \right)^9 - 3 \left(\frac{\sigma}{r} \right)^6 \right] \qquad r < r_c$$

r_c is the cutoff.

The *lj/class2/coul/cut* and *lj/class2/coul/long* styles add a Coulombic term as described for the *lj/cut* pair styles.

See *(Sun)* for a description of the COMPASS class2 force field.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff1 (distance units)
- cutoff2 (distance units)

The latter 2 coefficients are optional. If not specified, the global class 2 and Coulombic cutoffs are used. If only one cutoff is specified, it is used as the cutoff for both class 2 and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the class 2 and Coulombic cutoffs for this type pair. You cannot specify 2 cutoffs for style *lj/class2*, since it has no Coulombic terms.

For *lj/class2/coul/long* only the class 2 cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

If the pair_coeff command is not used to define coefficients for a particular I != J type pair, the mixing rule for ε and σ for all class2 potentials is to use the *sixthpower* formulas documented by the *pair_modify* command. The *pair_modify*

mix setting is thus ignored for class2 potentials for epsilon and sigma. However it is still followed for mixing the cutoff distance.

A version of these styles with a soft core, *lj/cut/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles. The version with soft core is only available if LAMMPS was built with that package. See the *Build package* page for more info.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.40.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/class2 pair styles can be mixed. Epsilon and sigma are always mixed with the value *sixthpower*. The cutoff distance is mixed by whatever option is set by the pair_modify command (default = geometric). See the "pair_modify" command for details.

All of the lj/class2 pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/class2/coul/long* pair style does not support the *pair_modify* table option since a tabulation capability has not yet been added to this potential.

All of the lj/class2 pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure of the Lennard-Jones portion of the pair interaction.

All of the lj/class2 pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

Only the *lj/class2* and *lj/class2/coul/long* pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. The other styles only support the *pair* keyword of run_style respa. See the *run_style* command for details.

4.40.5 Restrictions

These styles are part of the CLASS2 package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.40.6 Related commands

pair_coeff, *pair_style */soft*

4.40.7 Default

none

(Sun) Sun, J Phys Chem B 102, 7338-7364 (1998).

4.41 pair_style colloid command

Accelerator Variants: *colloid/gpu*, *colloid/omp*

4.41.1 Syntax

```
pair_style colloid cutoff
```

- cutoff = global cutoff for colloidal interactions (distance units)

4.41.2 Examples

```
pair_style colloid 10.0
pair_coeff * *  25 1.0 10.0 10.0
pair_coeff 1 1 144 1.0 0.0 0.0 3.0
pair_coeff 1 2  75.398 1.0 0.0 10.0 9.0
pair_coeff 2 2  39.478 1.0 10.0 10.0 25.0
```

4.41.3 Description

Style *colloid* computes pairwise interactions between large colloidal particles and small solvent particles using 3 formulas. A colloidal particle has a size > sigma; a solvent particle is the usual Lennard-Jones particle of size sigma.

The colloid-colloid interaction energy is given by

$$U_A = -\frac{A_{cc}}{6}\left[\frac{2a_1a_2}{r^2-(a_1+a_2)^2}+\frac{2a_1a_2}{r^2-(a_1-a_2)^2}+\ln\left(\frac{r^2-(a_1+a_2)^2}{r^2-(a_1-a_2)^2}\right)\right]$$

$$\begin{aligned}U_R =& \frac{A_{cc}}{37800}\frac{\sigma^6}{r}\left[\frac{r^2-7r(a_1+a_2)+6\left(a_1^2+7a_1a_2+a_2^2\right)}{(r-a_1-a_2)^7}\right.\\ &+\frac{r^2+7r(a_1+a_2)+6\left(a_1^2+7a_1a_2+a_2^2\right)}{(r+a_1+a_2)^7}\\ &-\frac{r^2+7r(a_1-a_2)+6\left(a_1^2-7a_1a_2+a_2^2\right)}{(r+a_1-a_2)^7}\\ &\left.-\frac{r^2-7r(a_1-a_2)+6\left(a_1^2-7a_1a_2+a_2^2\right)}{(r-a_1+a_2)^7}\right]\end{aligned}$$

$$U = U_A + U_R, \qquad r < r_c$$

where A_{cc} is the Hamaker constant, a_1 and a_2 are the radii of the two colloidal particles, and r_c is the cutoff. This equation results from describing each colloidal particle as an integrated collection of Lennard-Jones particles of size sigma and is derived in *(Everaers)*.

The colloid-solvent interaction energy is given by

$$U = \frac{2\,a^3\,\sigma^3\,A_{cs}}{9\left(a^2-r^2\right)^3}\left[1-\frac{\left(5\,a^6+45\,a^4\,r^2+63\,a^2\,r^4+15\,r^6\right)\sigma^6}{15\,(a-r)^6\,(a+r)^6}\right], \quad r < r_c$$

where A_{cs} is the Hamaker constant, a is the radius of the colloidal particle, and r_c is the cutoff. This formula is derived from the colloid-colloid interaction, letting one of the particle sizes go to zero.

The solvent-solvent interaction energy is given by the usual Lennard-Jones formula

$$U = \frac{A_{ss}}{36}\left[\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^{6}\right], \quad r < r_c$$

with A_{ss} set appropriately, which results from letting both particle sizes go to zero.

When used in combination with *pair_style yukawa/colloid*, the two terms become the so-called DLVO potential, which combines electrostatic repulsion and van der Waals attraction.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- A (energy units)
- σ (distance units)
- d1 (distance units)
- d2 (distance units)
- cutoff (distance units)

A is the Hamaker energy prefactor and should typically be set as follows:

- A_{cc} = colloid/colloid = $4\pi^2$ = 39.5
- A_{cs} = colloid/solvent = $\sqrt{A_{cc}A_{ss}}$

- A_{ss} = solvent/solvent = 144 (assuming epsilon = 1, so that 144/36 = 4)

σ is the size of the solvent particle or the constituent particles integrated over in the colloidal particle and should typically be set as follows:

- σ_{cc} = colloid/colloid = 1.0
- σ_{cs} = colloid/solvent = arithmetic mixing between colloid σ and solvent σ
- σ_{ss} = solvent/solvent = 1.0 or whatever size the solvent particle is

Thus typically σ_{cs} = 1.0, unless the solvent particle's size != 1.0.

D1 and d2 are particle diameters, so that d1 = 2*a1 and d2 = 2*a2 in the formulas above. Both d1 and d2 must be values >= 0. If d1 > 0 and d2 > 0, then the pair interacts via the colloid-colloid formula above. If d1 = 0 and d2 = 0, then the pair interacts via the solvent-solvent formula. I.e. a d value of 0 is a Lennard-Jones particle of size σ. If either d1 = 0 or d2 = 0 and the other is larger, then the pair interacts via the colloid-solvent formula.

Note that the diameter of a particular particle type may appear in multiple pair_coeff commands, as it interacts with other particle types. You should ensure the particle diameter is specified consistently each time it appears.

The last coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used. However, you typically want different cutoffs for interactions between different particle sizes. E.g. if colloidal particles of diameter 10 are used with solvent particles of diameter 1, then a solvent-solvent cutoff of 2.5 would correspond to a colloid-colloid cutoff of 25. A good rule-of-thumb is to use a colloid-solvent cutoff that is half the big diameter + 4 times the small diameter. I.e. 9 = 5 + 4 for the colloid-solvent cutoff in this case.

Note

When using pair_style colloid for a mixture with 2 (or more) widely different particles sizes (e.g. sigma=10 colloids in a background sigma=1 LJ fluid), you will likely want to use these commands for efficiency: *neighbor multi* and *comm_modify multi*.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.41.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A, sigma, d1, and d2 coefficients and cutoff distance for this pair style can be mixed. A is an energy value mixed like a LJ epsilon. D1 and d2 are distance values and are mixed like sigma. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.41.5 Restrictions

This style is part of the COLLOID package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Normally, this pair style should be used with finite-size particles which have a diameter, e.g. see the *atom_style sphere* command. However, this is not a requirement, since the only definition of particle size is via the pair_coeff parameters for each type. In other words, the physical radius of the particle is ignored. Thus you should ensure that the d1,d2 parameters you specify are consistent with the physical size of the particles of that type.

Per-particle polydispersity is not yet supported by this pair style; only per-type polydispersity is enabled via the pair_coeff parameters.

4.41.6 Related commands

pair_coeff

4.41.7 Default

none

(Everaers) Everaers, Ejtehadi, Phys Rev E, 67, 041710 (2003).

4.42 pair_style comb command

Accelerator Variants: *comb/omp*

4.43 pair_style comb3 command

4.43.1 Syntax

```
pair_style comb
pair_style comb3 keyword
```

keyword = polar
  polar value = polar_on or polar_off = whether or not to include atomic polarization

4.43.2 Examples

```
pair_style comb
pair_coeff * * ../potentials/ffield.comb Si
pair_coeff * * ../potentials/ffield.comb Hf Si O

pair_style comb3 polar_off
pair_coeff * * ../potentials/ffield.comb3 O Cu N C O
```

4.43.3 Description

Style *comb* computes the second generation variable charge COMB (Charge-Optimized Many-Body) potential. Style *comb3* computes the third-generation COMB potential. These COMB potentials are described in *(COMB)* and *(COMB3)*. Briefly, the total energy E_T of a system of atoms is given by

$$E_T = \sum_i [E_i^{self}(q_i) + \sum_{j>i} [E_{ij}^{short}(r_{ij}, q_i, q_j) + E_{ij}^{Coul}(r_{ij}, q_i, q_j)] +$$
$$E^{polar}(q_i, r_{ij}) + E^{vdW}(r_{ij}) + E^{barr}(q_i) + E^{corr}(r_{ij}, \theta_{jik})]$$

where E_i^{self} is the self-energy of atom i (including atomic ionization energies and electron affinities), E_{ij}^{short} is the bond-order potential between atoms i and j, E_{ij}^{Coul} is the Coulomb interactions, E^{polar} is the polarization term for organic systems (style *comb3* only), E^{vdW} is the van der Waals energy (style *comb3* only), E^{barr} is a charge barrier function, and E^{corr} are angular correction terms.

The COMB potentials (styles *comb* and *comb3*) are variable charge potentials. The equilibrium charge on each atom is calculated by the electronegativity equalization (QEq) method. See *Rick* for further details. This is implemented by the *fix qeq/comb* command, which should normally be specified in the input script when running a model with the COMB potential. The *fix qeq/comb* command has options that determine how often charge equilibration is performed, its convergence criterion, and which atoms are included in the calculation.

Only a single pair_coeff command is used with the *comb* and *comb3* styles which specifies the COMB potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the potential file in the pair_coeff command, where N is the number of LAMMPS atom types.

For example, if your LAMMPS simulation of a Si/SiO2/ HfO2 interface has 4 atom types, and you want the first and last to be Si, the second to be Hf, and the third to be O, and you would use the following pair_coeff command:

```
pair_coeff * * ../potentials/ffield.comb Si Hf O Si
```

The first two arguments must be * * so as to span all LAMMPS atom types. The first and last Si arguments map LAMMPS atom types 1 and 4 to the Si element in the *ffield.comb* file. The second Hf argument maps LAMMPS atom type 2 to the Hf element, and the third O argument maps LAMMPS atom type 3 to the O element in the potential file.

If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *comb* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

For style *comb*, the provided potential file *ffield.comb* contains all currently-available second generation COMB parameterizations: for Si, Cu, Hf, Ti, O, their oxides and Zr, Zn and U metals. For style *comb3*, the potential file *ffield.comb3* contains all currently-available third generation COMB parameterizations: O, Cu, N, C, H, Ti, Zn and Zr. The status of the optimization of the compounds, for example Cu2O, TiN and hydrocarbons, are given in the following table:

	O	Cu	N	C	H	Ti	Zn	Zr
O	F	F	F	F	F	F	F	F
Cu	F	F	P	F	F	P	F	P
N	F	P	F	M	F	P	P	P
C	F	F	M	F	F	M	M	M
H	F	F	F	F	F	M	M	F
Ti	F	P	P	M	M	F	P	P
Zn	F	F	P	M	M	P	F	P
Zr	F	P	P	M	F	P	P	F

- F = Fully optimized
- M = Only optimized for dimer molecule
- P = in progress, but have it from mixing rule

For style *comb3*, in addition to ffield.comb3, a special parameter file, *lib.comb3*, that is exclusively used for C/O/H systems, will be automatically loaded if carbon atom is detected in LAMMPS input structure. This file must be in your working directory or in the directories listed in the environment variable LAMMPS_POTENTIALS, as described on the *pair_coeff* command doc page.

The keyword *polar* indicates whether the force field includes the atomic polarization. Since the equilibration of the polarization has not yet been implemented, it can only set polar_off at present.

Note

You can not use potential file *ffield.comb* with style *comb3*, nor file *ffield.comb3* with style *comb*.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.43.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

These pair styles does not support the *pair_modify* shift, table, and tail options.

These pair styles do not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style, pair_coeff, and *fix qeq/comb* commands in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.43.5 Restrictions

These pair styles are part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles requires the *newton* setting to be "on" for pair interactions.

The COMB potentials in the *ffield.comb* and *ffield.comb3* files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the COMB potential with any LAMMPS units, but you would need to create your own COMB potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.43.6 Related commands

pair_style, *pair_coeff*, *fix qeq/comb*

4.43.7 Default

none

(COMB) T.-R. Shan, B. D. Devine, T. W. Kemper, S. B. Sinnott, and S. R. Phillpot, Phys. Rev. B 81, 125328 (2010)

(COMB3) T. Liang, T.-R. Shan, Y.-T. Cheng, B. D. Devine, M. Noordhoek, Y. Li, Z. Lu, S. R. Phillpot, and S. B. Sinnott, Mat. Sci. & Eng: R 74, 255-279 (2013).

(Rick) S. W. Rick, S. J. Stuart, B. J. Berne, J Chem Phys 101, 6141 (1994).

4.44 pair_style cosine/squared command

4.44.1 Syntax

```
pair_style cosine/squared cutoff
```

- cutoff = global cutoff for cosine-squared interactions (distance units)

```
pair_coeff I J eps sigma
pair_coeff I J eps sigma cutoff
pair_coeff I J eps sigma wca
pair_coeff I J eps sigma cutoff wca
```

- I, J = a particle type
- eps = interaction strength, i.e. the depth of the potential minimum (energy units)
- sigma = distance of the potential minimum from 0
- cutoff = the cutoff distance for this pair type, if different from global (distance units)
- wca = if specified a Weeks-Chandler-Andersen potential (with eps strength and minimum at sigma) is added, otherwise not

4.44.2 Examples

```
pair_style cosine/squared 3.0
pair_coeff * * 1.0 1.3
pair_coeff 1 3 1.0 1.3 2.0
pair_coeff 1 3 1.0 1.3 wca
pair_coeff 1 3 1.0 1.3 2.0 wca
```

4.44.3 Description

Style *cosine/squared* computes a potential of the form

$$E = \begin{cases} -\varepsilon & r < \sigma \\ -\varepsilon \cos\left(\frac{\pi(r-\sigma)}{2(r_c-\sigma)}\right)^2 & \sigma \le r < r_c \\ 0 & r \ge r_c \end{cases}$$

between two point particles, where $(\sigma, -\varepsilon)$ is the location of the (rightmost) minimum of the potential, as explained in the syntax section above.

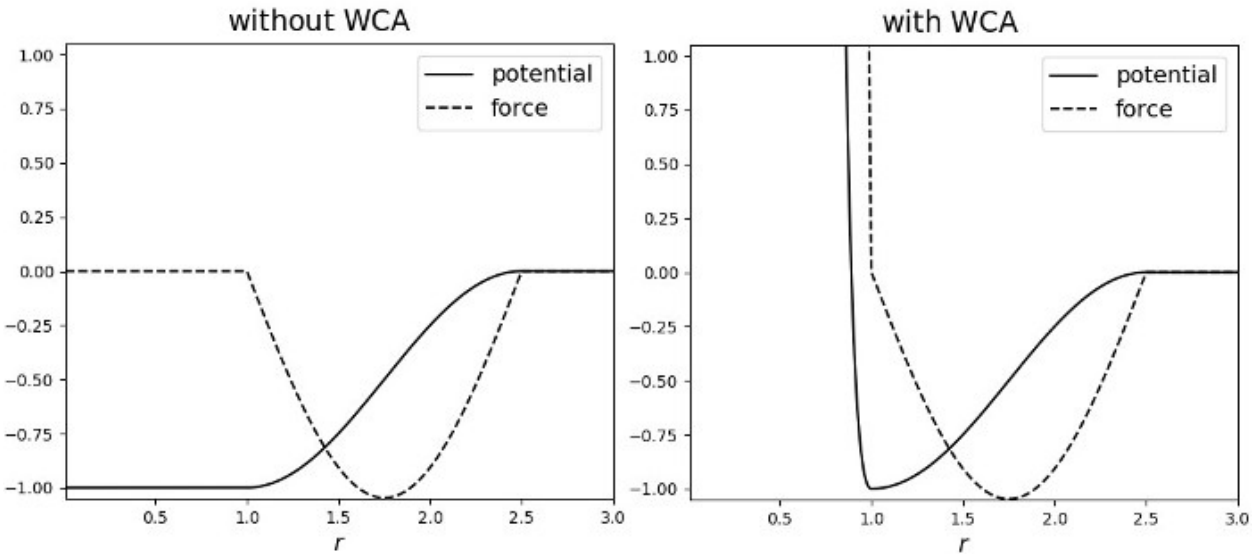
This potential was first used in *(Cooke)* for a coarse-grained lipid membrane model. It is generally very useful as a non-specific interaction potential because it is fully adjustable in depth and width while joining the minimum at (sigma, -epsilon) and zero at (cutoff, 0) smoothly, requiring no shifting and causing no related artifacts, tail energy calculations etc. This evidently requires *cutoff* to be larger than *sigma*.

If the *wca* option is used then a Weeks-Chandler-Andersen potential *(Weeks)* is added to the above specified cosine-squared potential, specifically the following:

$$E = \varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - 2\left(\frac{\sigma}{r}\right)^{6} + 1 \right], \quad r < \sigma$$

In this case, and this case only, the σ parameter can be equal to *cutoff* (σ = cutoff) which will result in ONLY the WCA potential being used (and print a warning), so the minimum will be attained at (sigma, 0). This is a convenience feature that enables a purely repulsive potential to be used without a need to define an additional pair style and use the hybrid styles.

The energy and force of this pair style for parameters epsilon = 1.0, sigma = 1.0, cutoff = 2.5, with and without the WCA potential, are shown in the graphs below:



4.44.4 Mixing, shift, table, tail correction, restart, rRESPA info

Mixing is not supported for this style.

The *shift*, *table* and *tail* options are not relevant for this style.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.44.5 Restrictions

The *cosine/squared* style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS is build with that package. See the *Build package* page for more info.

4.44.6 Related commands

pair_coeff , *pair_style lj/cut*

4.44.7 Default

none

(Cooke) "Cooke, Kremer and Deserno, Phys. Rev. E, 72, 011506 (2005)"

(Weeks) "Weeks, Chandler and Andersen, J. Chem. Phys., 54, 5237 (1971)"

4.45 pair_style coul/cut command

Accelerator Variants: *coul/cut/gpu*, *coul/cut/kk*, *coul/cut/omp*

4.46 pair_style coul/cut/global command

Accelerator Variants: *coul/cut/omp*

4.47 pair_style coul/debye command

Accelerator Variants: *coul/debye/gpu*, *coul/debye/kk*, *coul/debye/omp*

4.48 pair_style coul/dsf command

Accelerator Variants: *coul/dsf/gpu*, *coul/dsf/kk*, *coul/dsf/omp*

4.49 pair_style coul/exclude command

4.50 pair_style coul/long command

Accelerator Variants: *coul/long/omp*, *coul/long/kk*, *coul/long/gpu*

4.51 pair_style coul/msm command

Accelerator Variants: *coul/msm/omp*

4.52 pair_style coul/streitz command

4.53 pair_style coul/wolf command

Accelerator Variants: *coul/wolf/kk*, *coul/wolf/omp*

4.54 pair_style tip4p/cut command

Accelerator Variants: *tip4p/cut/omp*

4.55 pair_style tip4p/long command

Accelerator Variants: *tip4p/long/omp*

4.55.1 Syntax

```
pair_style coul/cut cutoff
pair_style coul/cut/global cutoff
pair_style coul/debye kappa cutoff
pair_style coul/dsf alpha cutoff
pair_style coul/exclude cutoff
pair_style coul/long cutoff
pair_style coul/wolf alpha cutoff
pair_style coul/streitz cutoff keyword alpha

* cutoff = global cutoff for Coulombic interactions
* kappa = Debye length (inverse distance units)
* alpha = damping parameter (inverse distance units)
```

```
pair_style tip4p/cut otype htype btype atype qdist cutoff
pair_style tip4p/long otype htype btype atype qdist cutoff

* otype,htype = atom types (numeric or type label) for TIP4P O and H
* btype,atype = bond and angle types (numeric or type label) for TIP4P waters
* qdist = distance from O atom to massless charge (distance units)
```

4.55.2 Examples

```
pair_style coul/cut 2.5
pair_coeff * *
pair_coeff 2 2 3.5

pair_style coul/debye 1.4 3.0
pair_coeff * *
pair_coeff 2 2 3.5

pair_style coul/dsf 0.05 10.0
pair_coeff * *

pair_style hybrid/overlay coul/exclude 10.0 ...
pair_coeff * * coul/exclude

pair_style coul/long 10.0
pair_coeff * *
```

(continues on next page)

(continued from previous page)

```
pair_style coul/msm 10.0
pair_coeff * *

pair_style coul/wolf 0.2 9.0
pair_coeff * *

pair_style coul/streitz 12.0 ewald
pair_style coul/streitz 12.0 wolf 0.30
pair_coeff * * AlO.streitz Al O

pair_style tip4p/cut 1 2 7 8 0.15 12.0
pair_coeff * *

pair_style tip4p/long 1 2 7 8 0.15 10.0
pair_coeff * *

pair_style tip4p/cut OW HW HW-OW HW-OW-HW 0.15 12.0
labelmap atom 1 OW 2 HW
labelmap bond 1 HW-OW
labelmap angle 1 HW-OW-HW
pair_coeff * *
```

4.55.3 Description

The *coul/cut* style computes the standard Coulombic interaction potential given by

$$E = \frac{Cq_iq_j}{\varepsilon r} \qquad r < r_c$$

where C is an energy-conversion constant, Qi and Qj are the charges on the two atoms, and ε is the dielectric constant which can be set by the *dielectric* command. The cutoff r_c truncates the interaction distance.

Pair style *coul/cut/global* computes the same Coulombic interactions as style *coul/cut* except that it allows only a single global cutoff and thus makes it compatible for use in combination with long-range coulomb styles in *hybrid pair styles*.

Style *coul/debye* adds an additional exp() damping factor to the Coulombic term, given by

$$E = \frac{Cq_iq_j}{\varepsilon r} \exp(-\kappa r) \qquad r < r_c$$

where κ is the Debye length. This potential is another way to mimic the screening effect of a polar solvent.

Style *coul/dsf* computes Coulombic interactions via the damped shifted force model described in *Fennell*, given by:

$$E = q_iq_j \left[\frac{\mathrm{erfc}(\alpha r)}{r} - \frac{\mathrm{erfc}(\alpha r_c)}{r_c} + \left(\frac{\mathrm{erfc}(\alpha r_c)}{r_c^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r_c^2)}{r_c} \right) (r - r_c) \right] \qquad r < r_c$$

where α is the damping parameter and *erfc()* is the complementary error-function. The potential corrects issues in the Wolf model (described below) to provide consistent forces and energies (the Wolf potential is not differentiable at the cutoff) and smooth decay to zero.

Style *coul/wolf* computes Coulombic interactions via the Wolf summation method, described in *Wolf*, given by:

$$E_i = \frac{1}{2}\sum_{j \neq i} \frac{q_i q_j \mathrm{erfc}(\alpha r_{ij})}{r_{ij}} + \frac{1}{2}\sum_{j \neq i} \frac{q_i q_j \mathrm{erf}(\alpha r_{ij})}{r_{ij}} \qquad r < r_c$$

where α is the damping parameter, and *erf()* and *erfc()* are error-function and complementary error-function terms. This potential is essentially a short-range, spherically-truncated, charge-neutralized, shifted, pairwise *1/r* summation. With a manipulation of adding and subtracting a self term (for i = j) to the first and second term on the right-hand-side, respectively, and a small enough α damping parameter, the second term shrinks and the potential becomes a rapidly-converging real-space summation. With a long enough cutoff and small enough α parameter, the energy and forces calculated by the Wolf summation method approach those of the Ewald sum. So it is a means of getting effective long-range interactions with a short-range potential.

Style *coul/streitz* is the Coulomb pair interaction defined as part of the Streitz-Mintmire potential, as described in *this paper*, in which charge distribution about an atom is modeled as a Slater 1*s* orbital. More details can be found in the referenced paper. To fully reproduce the published Streitz-Mintmire potential, which is a variable charge potential, style *coul/streitz* must be used with *pair_style eam/alloy* (or some other short-range potential that has been parameterized appropriately) via the *pair_style hybrid/overlay* command. Likewise, charge equilibration must be performed via the *fix qeq/slater* command. For example:

```
pair_style hybrid/overlay coul/streitz 12.0 wolf 0.31 eam/alloy
pair_coeff * * coul/streitz AlO.streitz Al O
pair_coeff * * eam/alloy AlO.eam.alloy Al O
fix 1 all qeq/slater 1 12.0 1.0e-6 100 coul/streitz
```

The keyword *wolf* in the coul/streitz command denotes computing Coulombic interactions via Wolf summation. An additional damping parameter is required for the Wolf summation, as described for the coul/wolf potential above. Alternatively, Coulombic interactions can be computed via an Ewald summation. For example:

```
pair_style hybrid/overlay coul/streitz 12.0 ewald eam/alloy
kspace_style ewald 1e-6
```

Keyword *ewald* does not need a damping parameter, but a *kspace_style* must be defined, which can be style *ewald* or *pppm*. The Ewald method was used in Streitz and Mintmire's original paper, but a Wolf summation offers a speed-up in some cases.

For the fix qeq/slater command, the *qfile* can be a filename that contains QEq parameters as discussed on the *fix qeq* command doc page. Alternatively *qfile* can be replaced by "coul/streitz", in which case the fix will extract QEq parameters from the coul/streitz pair style itself.

See the examples/strietz directory for an example input script that uses the Streitz-Mintmire potential. The potentials directory has the AlO.eam.alloy and AlO.streitz potential files used by the example.

Note that the Streiz-Mintmire potential is generally used for oxides, but there is no conceptual problem with extending it to nitrides and carbides (such as SiC, TiN). Pair coul/strietz used by itself or with any other pair style such as EAM, MEAM, Tersoff, or LJ in hybrid/overlay mode. To do this, you would need to provide a Streitz-Mintmire parameterization for the material being modeled.

Pair style *coul/exclude* computes Coulombic interactions like *coul/cut* but **only** applies them to excluded pairs using a scaling factor of $\gamma - 1.0$ with γ being the factor assigned to that excluded pair via the *special_bonds coul* setting. With this it is possible to treat Coulomb interactions for molecular systems with *kspace style scafacos*, which always computes the *full* Coulomb interactions without exclusions. Pair style *coul/exclude* will then *subtract* the excluded interactions accordingly. So to achieve the same forces as with pair_style lj/cut/coul/long 12.0 with kspace_style

pppm 1.0e-6, one would use pair_style hybrid/overlay lj/cut 12.0 coul/exclude 12.0 with kspace_style scafacos p3m 1.0e-6.

Styles *coul/long* and *coul/msm* compute the same Coulombic interactions as style *coul/cut* except that an additional damping factor is applied so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

Styles *tip4p/cut* and *tip4p/long* implement the Coulomb part of the TIP4P water model of *(Jorgensen)*, which introduces a massless site located a short distance away from the oxygen atom along the bisector of the HOH angle. The atomic types of the oxygen and hydrogen atoms, the bond and angle types for OH and HOH interactions, and the distance to the massless charge site are specified as pair_style arguments. Style *tip4p/cut* uses a global cutoff for Coulomb interactions; style *tip4p/long* is for use with a long-range Coulombic solver (Ewald or PPPM).

Note

For each TIP4P water molecule in your system, the atom IDs for the O and 2 H atoms must be consecutive, with the O atom first. This is to enable LAMMPS to "find" the 2 H atoms associated with each O atom. For example, if the atom ID of an O atom in a TIP4P water molecule is 500, then its 2 H atoms must have IDs 501 and 502.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

See the *Howto tip4p* page for more information on how to use the TIP4P pair styles and lists of parameters to set. Note that the neighbor list cutoff for Coulomb interactions is effectively extended by a distance 2*qdist when using the TIP4P pair style, to account for the offset distance of the fictitious charges on O atoms in water molecules. Thus it is typically best in an efficiency sense to use a LJ cutoff >= Coulombic cutoff + 2*qdist, to shrink the size of the neighbor list. This leads to slightly larger cost for the long-range calculation, so you can test the trade-off for your model.

Note that these potentials are designed to be combined with other pair potentials via the *pair_style hybrid/overlay* command. This is because they have no repulsive core. Hence if they are used by themselves, there will be no repulsion to keep two oppositely charged particles from moving arbitrarily close to each other.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutoff (distance units)

For *coul/cut* and *coul/debye* the cutoff coefficient is optional. If it is not used (as in some of the examples above), the default global value specified in the pair_style command is used.

For *coul/cut/global*, *coul/long* and *coul/msm* no cutoff can be specified for an individual I,J type pair via the pair_coeff command. All type pairs use the same global Coulomb cutoff specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.55.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the cutoff distance for the *coul/cut* style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

The *pair_modify* shift option is not relevant for these pair styles.

The *coul/long* style supports the *pair_modify* table option for tabulation of the short-range portion of the long-range Coulombic interaction.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.55.5 Restrictions

The *coul/long*, *coul/msm*, *coul/streitz*, and *tip4p/long* styles are part of the KSPACE package. The *coul/cut/global*, *coul/exclude* styles are part of the EXTRA-PAIR package. The *tip4p/cut* style is part of the MOLECULE package. A pair style is only enabled if LAMMPS was built with its corresponding package. See the *Build package* doc page for more info.

4.55.6 Related commands

pair_coeff , *pair_style, hybrid/overlay*, *kspace_style*

4.55.7 Default

none

(Wolf) D. Wolf, P. Keblinski, S. R. Phillpot, J. Eggebrecht, J Chem Phys, 110, 8254 (1999).

(Fennell) C. J. Fennell, J. D. Gezelter, J Chem Phys, 124, 234104 (2006).

(Streitz) F. H. Streitz, J. W. Mintmire, Phys Rev B, 50, 11996-12003 (1994).

(Jorgensen) Jorgensen, Chandrasekhar, Madura, Impey, Klein, J Chem Phys, 79, 926 (1983).

4.56 pair_style coul/diel command

Accelerator Variants: *coul/diel/omp*

4.56.1 Syntax

```
pair_style coul/diel cutoff
```

cutoff = global cutoff (distance units)

4.56.2 Examples

```
pair_style coul/diel 3.5
pair_coeff 1 4 78. 1.375 0.112
```

4.56.3 Description

Style *coul/diel* computes a Coulomb correction for implicit solvent ion interactions in which the dielectric permittivity is distance dependent. The dielectric permittivity $\varepsilon_D(r)$ connects to limiting regimes: One limit is defined by a small dielectric permittivity (close to vacuum) at or close to contact separation between the ions. At larger separations the dielectric permittivity reaches a bulk value used in the regular Coulomb interaction coul/long or coul/cut. The transition is modeled by a hyperbolic function which is incorporated in the Coulomb correction term for small ion separations as follows

$$E = \frac{Cq_iq_j}{\varepsilon r}\left(\frac{\varepsilon}{\varepsilon_D(r)} - 1\right) \qquad r < r_c$$
$$\varepsilon_D(r) = \frac{5.2+\varepsilon}{2} + \frac{\varepsilon - 5.2}{2}\tanh\left(\frac{r - r_{me}}{\sigma_e}\right)$$

where r_{me} is the inflection point of $\varepsilon_D(r)$ and σ_e is a slope defining length scale. C is the same Coulomb conversion factor as in the pair_styles coul/cut, coul/long, and coul/debye. In this way the Coulomb interaction between ions is corrected at small distances r. The lower limit of $\varepsilon_D(r \to 0) = 5.2$ due to dielectric saturation *(Stiles)* while the Coulomb interaction reaches its bulk limit by setting $\varepsilon_D(r \to \infty) = \varepsilon$, the bulk value of the solvent which is 78 for water at 298K.

Examples of the use of this type of Coulomb interaction include implicit solvent simulations of salt ions *(Lenart)* and of ionic surfactants *(Jusufi)*. Note that this potential is only reasonable for implicit solvent simulations and in combination with coul/cut or coul/long. It is also usually combined with gauss/cut, see *(Lenart)* or *(Jusufi)*.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- ε (no units)
- r_{me} (distance units)
- σ_e (distance units)

The global cutoff (r_c) specified in the pair_style command is used.

4.56.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support parameter mixing. Coefficients must be given explicitly for each type of particle pairs.

This pair style supports the *pair_modify* shift option for the energy of the Gauss-potential portion of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.56.5 Restrictions

This style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.56.6 Related commands

pair_coeff *pair_style gauss/cut*

4.56.7 Default

none

(Stiles) Stiles, Hubbard, and Kayser, J Chem Phys, 77, 6189 (1982).

(Lenart) Lenart, Jusufi, and Panagiotopoulos, J Chem Phys, 126, 044509 (2007).

(Jusufi) Jusufi, Hynninen, and Panagiotopoulos, J Phys Chem B, 112, 13783 (2008).

4.57 pair_style coul/shield command

4.57.1 Syntax

```
pair_style coul/shield cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.57.2 Examples

```
pair_style coul/shield 16.0 1
pair_coeff 1 2 0.70
```

4.57.3 Description

Style *coul/shield* computes a Coulomb interaction for boron and nitrogen atoms located in different layers of hexagonal boron nitride. This potential is designed be used in combination with the pair style *ilp/graphene/hbn*

Note

This potential is intended for electrostatic interactions between two different layers of hexagonal boron nitride. Therefore, to avoid interaction within the same layers, each layer should have a separate molecule id and is recommended to use the "full" atom style, so that charge and molecule ID information is included.

$$E = \frac{1}{2}\sum_{i}\sum_{j\neq i} V_{ij}$$

$$V_{ij} = \mathrm{Tap}(r_{ij})\frac{\kappa q_i q_j}{\sqrt[3]{r_{ij}^3 + (1/\lambda_{ij})^3}}$$

$$\mathrm{Tap}(r_{ij}) = 20\left(\frac{r_{ij}}{R_{cut}}\right)^7 - 70\left(\frac{r_{ij}}{R_{cut}}\right)^6 + 84\left(\frac{r_{ij}}{R_{cut}}\right)^5 - 35\left(\frac{r_{ij}}{R_{cut}}\right)^4 + 1$$

Where $\mathrm{Tap}(r_{ij})$ is the taper function which provides a continuous cutoff (up to third derivative) for inter-atomic separations larger than r_c *(Leven1)*, *(Leven2)* and *(Maaravi)*. Here λ is the shielding parameter that eliminates the short-range singularity of the classical mono-polar electrostatic interaction expression *(Maaravi)*.

The shielding parameter λ (1/distance units) must be defined for each pair of atom types via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

The global cutoff (r_c) specified in the pair_style command is used.

4.57.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support parameter mixing. Coefficients must be given explicitly for each type of particle pairs.

The *pair_modify table* option is not relevant for this pair style.

This pair style does not support the *pair_modify tail* option for adding long-range tail corrections to energy and pressure.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.57.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.57.6 Related commands

pair_coeff *pair_style ilp/graphene/hbn*

4.57.7 Default

tap_flag = 1

(Leven1) I. Leven, I. Azuri, L. Kronik and O. Hod, J. Chem. Phys. 140, 104106 (2014).

(Leven2) I. Leven et al, J. Chem.Theory Comput. 12, 2896-905 (2016).

(Maaravi) T. Maaravi et al, J. Phys. Chem. C 121, 22826-22835 (2017).

4.58 pair_style coul/slater command

4.59 pair_style coul/slater/cut command

4.60 pair_style coul/slater/long command

Accelerator Variants: *coul/slater/long/gpu*

4.60.1 Syntax

```
pair_style coul/slater/cut lambda cutoff
pair_style coul/slater/long lambda cutoff
```

lambda = decay length of the charge (distance units) cutoff = cutoff (distance units)

4.60.2 Examples

```
pair_style coul/slater/cut 1.0 3.5
pair_coeff * *
pair_coeff 2 2 2.5

pair_style coul/slater/long 1.0 12.0
pair_coeff * *
pair_coeff 1 1 5.0
```

4.60.3 Description

Styles *coul/slater/** compute electrostatic interactions in mesoscopic models which employ potentials without explicit excluded-volume interactions. The goal is to prevent artificial ionic pair formation by including a charge distribution in the Coulomb potential, following the formulation of *(Melchor)*:

$$E = \frac{Cq_iq_j}{\varepsilon r}\left(1 - \left(1 + \frac{r_{ij}}{\lambda} exp\left(-2r_{ij}/\lambda\right)\right)\right) \qquad r < r_c$$

where r_c is the cutoff distance and λ is the decay length of the charge. C is the same Coulomb conversion factor as in the pair_styles coul/cut and coul/long. In this way the Coulomb interaction between ions is corrected at small distances r. For the *coul/slater/cut* style, the potential energy for distances larger than the cutoff is zero, while for the *coul/slater/long*, the long-range interactions are computed either by the Ewald or the PPPM technique.

Phenomena that can be captured at a mesoscopic level using this type of electrostatic interactions include the formation of polyelectrolyte-surfactant aggregates, charge stabilization of colloidal suspensions, and the formation of complexes driven by charged species in biological systems. *(Vaiwala)*.

The cutoff distance is optional. If it is not used, the default global value specified in the pair_style command is used. For each pair of atom types, a specific cutoff distance can be defined via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- r_c (distance units)

The global decay length of the charge (λ) specified in the pair_style command is used for all pairs.

4.60.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the cutoff distance for the *coul/slater* styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

The *pair_modify* shift and table options are not relevant for these pair styles.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.60.5 Restrictions

The *coul/slater/long* style requires the long-range solvers included in the KSPACE package.

These styles are part of the EXTRA-PAIR package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.60.6 Related commands

pair_coeff, *pair_style, hybrid/overlay*, *kspace_style*

4.60.7 Default

none

(Melchor) Gonzalez-Melchor, Mayoral, Velazquez, and Alejandre, J Chem Phys, 125, 224107 (2006).

(Vaiwala) Vaiwala, Jadhav, and Thaokar, J Chem Phys, 146, 124904 (2017).

4.61 pair_style coul/tt command

4.61.1 Syntax

```
pair_style style args
```

- style = *coul/tt*
- args = list of arguments for a particular style

coul/tt args = n cutoff
  n = degree of polynomial
  cutoff = global cutoff (distance units)

4.61.2 Examples

```
pair_style hybrid/overlay ... coul/tt 4 12.0
pair_coeff 1 2  coul/tt 4.5 1.0
pair_coeff 1 2  coul/tt 4.0 1.0 4 12.0
pair_coeff 1 3* coul/tt 4.5 1.0 4
```

Example input scripts available: examples/PACKAGES/drude

4.61.3 Description

The *coul/tt* pair style is meant to be used with force fields that include explicit polarization through Drude dipoles.

The *coul/tt* pair style should be used as a sub-style within in the *pair_style hybrid/overlay* command, in conjunction with a main pair style including Coulomb interactions and *thole* pair style, or with *lj/cut/thole/long* pair style that is equivalent to the combination of preceding two.

The *coul/tt* pair styles compute the charge-dipole Coulomb interaction damped at short distances by a function

$$f_{n,ij}(r) = 1 - c_{ij} \cdot e^{-b_{ij}r} \sum_{k=0}^{n} \frac{(b_{ij}r)^k}{k!}$$

This function results from an adaptation to the Coulomb interaction *(Salanne)* of the damping function originally proposed by *Tang Toennies* for van der Waals interactions.

The polynomial takes the degree 4 for damping the Coulomb interaction. The parameters b_{ij} and c_{ij} could be determined from first-principle calculations for small, mainly mono-atomic, ions *(Salanne)*, or else treated as empirical for large molecules.

In pair styles with Drude induced dipoles, this damping function is typically applied to the interactions between a Drude charge (either $q_{D,i}$ on a Drude particle or $-q_{D,i}$ on the respective Drude core)) and a charge on a non-polarizable atom, q_j.

The Tang-Toennies function could also be used to damp electrostatic interactions between the (non-polarizable part of the) charge of a core, $q_i - q_{D,i}$, and the Drude charge of another, $-q_{D,j}$. The b_{ij} and c_{ij} are equal to b_{ji} and c_{ji} in the case of core-core interactions.

For pair_style *coul/tt*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the example above.

- b_{ij}
- c_{ij}
- degree of polynomial (positive integer)
- cutoff (distance units)

The last two coefficients are optional. If not specified the global degree of the polynomial or the global cutoff specified in the pair_style command are used. In order to specify a cutoff (forth argument), the degree of the polynomial (third argument) must also be specified.

4.61.4 Mixing, shift, table, tail correction, restart, rRESPA info

The *coul/tt* pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

4.61.5 Restrictions

These pair styles are part of the DRUDE package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair_style should currently not be used with the *charmm dihedral style* if the latter has non-zero 1-4 weighting factors. This is because the *coul/tt* pair style does not know which pairs are 1-4 partners of which dihedrals.

4.61.6 Related commands

fix drude, *fix langevin/drude*, *fix drude/transform*, *compute temp/drude*, *pair_style thole*

4.61.7 Default

none

(Thole) Chem Phys, 59, 341 (1981).

(Salanne) Salanne, Rotenberg, Jahn, Vuilleumier, Simon, Christian and Madden, Theor Chem Acc, 131, 1143 (2012).

(Tang and Toennies) J Chem Phys, 80, 3726 (1984).

4.62 pair_style born/coul/dsf/cs command

4.63 pair_style born/coul/long/cs command

Accelerator Variants: *born/coul/long/cs/gpu*

4.64 pair_style born/coul/wolf/cs command

Accelerator Variants: *born/coul/wolf/cs/gpu*

4.65 pair_style buck/coul/long/cs command

4.66 pair_style coul/long/cs command

Accelerator Variants: *coul/long/cs/gpu*

4.67 pair_style coul/wolf/cs command

4.68 pair_style lj/cut/coul/long/cs command

4.69 pair_style lj/class2/coul/long/cs command

4.69.1 Syntax

```
pair_style style args
```

- style = *born/coul/dsf/cs* or *born/coul/long/cs* or *born/coul/wolf/cs* or *buck/coul/long/cs* or *coul/long/cs* or *coul/wolf/cs* or *lj/cut/coul/long/cs* or *lj/class2/coul/long/cs*
- args = list of arguments for a particular style

born/coul/dsf/cs args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (distance units)
born/coul/long/cs args = cutoff (cutoff2)
  cutoff = global cutoff for non-Coulombic (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
born/coul/wolf/cs args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
buck/coul/long/cs args = cutoff (cutoff2)
  cutoff = global cutoff for Buckingham (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

coul/long args = cutoff
  cutoff = global cutoff for Coulombic (distance units)
coul/wolf args = alpha cutoff
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for Coulombic (distance units)
lj/cut/coul/long/cs args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/class2/coul/long/cs args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.69.2 Examples

```
pair_style born/coul/dsf/cs 0.1 10.0 12.0
pair_coeff * *   0.0 1.00 0.00 0.00 0.00
pair_coeff 1 1 480.0 0.25 0.00 1.05 0.50

pair_style born/coul/long/cs 10.0 8.0
pair_coeff 1 1 6.08 0.317 2.340 24.18 11.51

pair_style born/coul/wolf/cs 0.25 10.0 12.0
pair_coeff * *   0.0 1.00 0.00 0.00 0.00
pair_coeff 1 1 480.0 0.25 0.00 1.05 0.50

pair_style buck/coul/long/cs 10.0
pair_style buck/coul/long/cs 10.0 8.0
pair_coeff * * 100.0 1.5 200.0
pair_coeff 1 1 100.0 1.5 200.0 9.0

pair_style coul/long/cs 10.0
pair_coeff * *

pair_style coul/wolf/cs 0.2 9.0
pair_coeff * *

pair_style lj/cut/coul/long/cs 10.0
pair_style lj/cut/coul/long/cs 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0
```

4.69.3 Description

These pair styles are designed to be used with the adiabatic core/shell model of *(Mitchell and Fincham)*. See the *Howto coreshell* page for an overview of the model as implemented in LAMMPS.

All the styles are identical to the corresponding pair style without the "/cs" in the name:

- *pair_style born/coul/dsf*
- *pair_style born/coul/long*
- *pair_style born/coul/wolf*

- *pair_style buck/coul/long*
- *pair_style coul/long*
- *pair_style coul/wolf*
- *pair_style lj/cut/coul/long*
- *pair_style lj/class2/coul/long*

except that they correctly treat the special case where the distance between two charged core and shell atoms in the same core/shell pair approach r = 0.0.

Styles with a "/long" in the name are used with a long-range solver for Coulombic interactions via the *kspace_style* command. They require special treatment of the short-range Coulombic interactions within the cor/shell model.

Specifically, the short-range Coulomb interaction between a core and its shell should be turned off using the *special_bonds* command by setting the 1-2 weight to 0.0, which works because the core and shell atoms are bonded to each other. This induces a long-range correction approximation which fails at small distances (~< 10e-8). Therefore, the Coulomb term which is used to calculate the correction factor is extended by a minimal distance (r_min = 1.0-6) when the interaction between a core/shell pair is treated, as follows

$$E = \frac{Cq_iq_j}{\varepsilon(r + r_{min})} \qquad r \to 0$$

where C is an energy-conversion constant, q_i and q_j are the charges on the core and shell, epsilon is the dielectric constant and r_{min} is the minimal distance.

For styles that are not used with a long-range solver, i.e. those with "/dsf" or "/wolf" in the name, the only correction is the addition of a minimal distance to avoid the possible r = 0.0 case for a core/shell pair.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.69.4 Mixing, shift, table, tail correction, restart, rRESPA info

See the corresponding doc pages for pair styles without the "cs" suffix to see how mixing, shifting, tabulation, tail correction, restarting, and rRESPA are handled by theses pair styles.

4.69.5 Restrictions

These pair styles are part of the CORESHELL package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.69.6 Related commands

pair_coeff, *pair_style born*, *pair_style buck*

4.69.7 Default

none

(Mitchell and Fincham) Mitchell, Fincham, J Phys Condensed Matter, 5, 1031-1038 (1993).

4.70 pair_style coul/cut/dielectric command

4.71 pair_style coul/long/dielectric command

4.72 pair_style lj/cut/coul/cut/dielectric command

Accelerator Variants: *lj/cut/coul/cut/dielectric/omp*

4.73 pair_style lj/cut/coul/debye/dielectric command

Accelerator Variants: *lj/cut/coul/debye/dielectric/omp*

4.74 pair_style lj/cut/coul/long/dielectric command

Accelerator Variants: *lj/cut/coul/long/dielectric/omp*

4.75 pair_style lj/cut/coul/msm/dielectric command

4.76 pair_style lj/long/coul/long/dielectric command

4.76.1 Syntax

```
pair_style style args
```

- style = *lj/cut/coul/cut/dielectric* or *lj/cut/coul/long/dielectric* or *lj/cut/coul/msm/dielectric* or *lj/long/coul/msm/dielectric*
- args = list of arguments for a particular style

4.76.2 Examples

```
pair_style coul/cut/dielectric 10.0
pair_coeff * *
pair_coeff 1 1 9.0

pair_style lj/cut/coul/cut/dielectric 10.0
pair_style lj/cut/coul/cut/dielectric 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/cut/coul/long/dielectric 10.0
pair_style lj/cut/coul/long/dielectric 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0
```

Used in input scripts:

```
examples/PACKAGES/dielectric/in.confined
examples/PACKAGES/dielectric/in.nopbc
```

4.76.3 Description

All these pair styles are derived from the corresponding pair styles without the *dielectric* suffix. In addition to computing atom forces and energies, these pair styles compute the electric field vector at each atom, which are intended to be used by the *fix polarize* commands to compute induced charges at interfaces between two regions of different dielectric constant.

These pair styles should be used with *atom_style dielectric*.

The styles lj/cut/coul/long/dielectric, lj/cut/coul/msm/dielectric, and lj/long/coul/long/dielectric should be used with their kspace style counterparts, namely, pppm/dielectric, pppm/disp/dielectric, and msm/dielectric, respectively.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.76.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distances for this pair style can be mixed. The default mix algorithm is *geometric*. See the *pair_modify*" command for details.

The *pair_modify* table option is not relevant for this pair style.

These pair styles write its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.76.5 Restrictions

These styles are part of the DIELECTRIC package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.76.6 Related commands

pair_coeff , *fix polarize*, *read_data*

4.76.7 Default

none

4.77 pair_style lj/cut/dipole/cut command

Accelerator Variants: *lj/cut/dipole/cut/gpu*, *lj/cut/dipole/cut/kk*, *lj/cut/dipole/cut/omp*

4.78 pair_style lj/sf/dipole/sf command

Accelerator Variants: *lj/sf/dipole/sf/gpu*, *lj/sf/dipole/sf/omp*

4.79 pair_style lj/cut/dipole/long command

Accelerator Variants: *lj/cut/dipole/long/gpu*

4.80 pair_style lj/long/dipole/long command

4.80.1 Syntax

```
pair_style lj/cut/dipole/cut cutoff (cutoff2)
pair_style lj/sf/dipole/sf cutoff (cutoff2)
pair_style lj/cut/dipole/long cutoff (cutoff2)
pair_style lj/long/dipole/long flag_lj flag_coul cutoff (cutoff2)
```

- cutoff = global cutoff LJ (and Coulombic if only 1 arg) (distance units)
- cutoff2 = global cutoff for Coulombic and dipole (optional) (distance units)
- flag_lj = *long* or *cut* or *off*

  long = use long-range damping on dispersion 1/r^6 term
  cut = use a cutoff on dispersion 1/r^6 term
  off = omit disperion 1/r^6 term entirely

- flag_coul = *long* or *off*

  long = use long-range damping on Coulombic 1/r and point-dipole terms
  off = omit Coulombic and point-dipole terms entirely

4.80.2 Examples

```
pair_style lj/cut/dipole/cut 2.5 5.0
pair_coeff * * 1.0 1.0
pair_coeff 2 3 0.8 1.0 2.5 4.0

pair_style lj/sf/dipole/sf 9.0
pair_coeff * * 1.0 1.0
pair_coeff 2 3 1.0 1.0 2.5 4.0 scale 0.5
pair_coeff 2 3 0.8 1.0 2.5 4.0

pair_style lj/cut/dipole/long 2.5 3.5
pair_coeff * * 1.0 1.0
pair_coeff 2 3 0.8 1.0 3.0

pair_style lj/long/dipole/long long long 3.5
pair_coeff * * 1.0 1.0
pair_coeff 2 3 0.8 1.0

pair_style lj/long/dipole/long cut long 2.5 3.5
pair_coeff * * 1.0 1.0
pair_coeff 2 3 0.8 1.0 3.0
```

4.80.3 Description

Style *lj/cut/dipole/cut* computes interactions between pairs of particles that each have a charge and/or a point dipole moment. In addition to the usual Lennard-Jones interaction between the particles (Elj) the charge-charge (Eqq), charge-dipole (Eqp), and dipole-dipole (Epp) interactions are computed by these formulas for the energy (E), force (F), and torque (T) between particles I and J.

$$
\begin{aligned}
E_{LJ} =& 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \\
E_{qq} =& \frac{q_i q_j}{r} \\
E_{qp} =& \frac{q}{r^3}(p \bullet \vec{r}) \\
E_{pp} =& \frac{1}{r^3}(\vec{p}_i \bullet \vec{p}_j) - \frac{3}{r^5}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \bullet \vec{r})
\end{aligned}
$$

$$
\begin{aligned}
F_{qq} =& \frac{q_i q_j}{r^3}\vec{r} \\
F_{qp} =& -\frac{q}{r^3}\vec{p} + \frac{3q}{r^5}(\vec{p} \bullet \vec{r})\vec{r} \\
F_{pp} =& \frac{3}{r^5}(\vec{p}_i \bullet \vec{p}_j)\vec{r} - \frac{15}{r^7}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \bullet \vec{r})\vec{r} + \frac{3}{r^5}\left[(\vec{p}_j \bullet \vec{r})\vec{p}_i + (\vec{p}_i \bullet \vec{r})\vec{p}_j\right]
\end{aligned}
$$

$$
\begin{aligned}
T_{pq} = T_{ij} =& \frac{q_j}{r^3}(\vec{p}_i \times \vec{r}) \\
T_{qp} = T_{ji} =& -\frac{q_i}{r^3}(\vec{p}_j \times \vec{r}) \\
T_{pp} = T_{ij} =& -\frac{1}{r^3}(\vec{p}_i \times \vec{p}_j) + \frac{3}{r^5}(\vec{p}_j \bullet \vec{r})(\vec{p}_i \times \vec{r}) \\
T_{pp} = T_{ji} =& -\frac{1}{r^3}(\vec{p}_j \times \vec{p}_i) + \frac{3}{r^5}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \times \vec{r})
\end{aligned}
$$

where q_i and q_j are the charges on the two particles, $\vec{p}_i$ and $\vec{p}_j$ are the dipole moment vectors of the two particles, r is their separation distance, and the vector r = Ri - Rj is the separation vector between the two particles. Note that Eqq and Fqq are simply Coulombic energy and force, Fij = -Fji as symmetric forces, and Tij != -Tji since the torques do not act symmetrically. These formulas are discussed in *(AllenTildesley)* and in *(Toukmaji)*.

Also note, that in the code, all of these terms (except Elj) have a C/ε prefactor, the same as the Coulombic term in the LJ + Coulombic pair styles discussed *here*. C is an energy-conversion constant and ε is the dielectric constant which can be set by the *dielectric* command. The same is true of the equations that follow for other dipole pair styles.

Style *lj/sf/dipole/sf* computes "shifted-force" interactions between pairs of particles that each have a charge and/or a point dipole moment. In general, a shifted-force potential is a (slightly) modified potential containing extra terms that make both the energy and its derivative go to zero at the cutoff distance; this removes (cutoff-related) problems in energy conservation and any numerical instability in the equations of motion *(AllenTildesley)*. Shifted-force interactions for the Lennard-Jones (E_LJ), charge-charge (Eqq), charge-dipole (Eqp), dipole-charge (Epq) and dipole-dipole (Epp)

potentials are computed by these formulas for the energy (E), force (F), and torque (T) between particles I and J:

$$E_{LJ} = 4\varepsilon \left\{ \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] + \left[6\left(\frac{\sigma}{r_c}\right)^{12} - 3\left(\frac{\sigma}{r_c}\right)^{6} \right] \left(\frac{r}{r_c}\right)^{2} - 7\left(\frac{\sigma}{r_c}\right)^{12} + 4\left(\frac{\sigma}{r_c}\right)^{6} \right\}$$

$$E_{qq} = \frac{q_i q_j}{r} \left(1 - \frac{r}{r_c}\right)^{2}$$

$$E_{pq} = E_{ji} = -\frac{q}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] (\vec{p} \bullet \vec{r})$$

$$E_{qp} = E_{ij} = \frac{q}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] (\vec{p} \bullet \vec{r})$$

$$E_{pp} = \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + 3\left(\frac{r}{r_c}\right)^{4} \right] \left[\frac{1}{r^3}(\vec{p}_i \bullet \vec{p}_j) - \frac{3}{r^5}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \bullet \vec{r}) \right]$$

$$F_{LJ} = \left\{ \left[48\varepsilon \left(\frac{\sigma}{r}\right)^{12} - 24\varepsilon \left(\frac{\sigma}{r}\right)^{6} \right] \frac{1}{r^2} - \left[48\varepsilon \left(\frac{\sigma}{r_c}\right)^{12} - 24\varepsilon \left(\frac{\sigma}{r_c}\right)^{6} \right] \frac{1}{r_c^2} \right\} \vec{r}$$

$$F_{qq} = \frac{q_i q_j}{r} \left(\frac{1}{r^2} - \frac{1}{r_c^2}\right) \vec{r}$$

$$F_{pq} = F_{ij} = -\frac{3q}{r^5} \left[1 - \left(\frac{r}{r_c}\right)^{2} \right] (\vec{p} \bullet \vec{r})\vec{r} + \frac{q}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] \vec{p}$$

$$F_{qp} = F_{ij} = \frac{3q}{r^5} \left[1 - \left(\frac{r}{r_c}\right)^{2} \right] (\vec{p} \bullet \vec{r})\vec{r} - \frac{q}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] \vec{p}$$

$$F_{pp} = \frac{3}{r^5} \left\{ \left[1 - \left(\frac{r}{r_c}\right)^{4} \right] \left[(\vec{p}_i \bullet \vec{p}_j) - \frac{3}{r^2}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \bullet \vec{r}) \right] \vec{r} + \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + 3\left(\frac{r}{r_c}\right)^{4} \right] \left[(\vec{p}_j \bullet \vec{r})\vec{p}_i + (\vec{p}_i \bullet \vec{r})\vec{p}_j - \frac{2}{r^2}(\vec{p}_i \bullet \vec{r})(\vec{p}_j \bullet \vec{r})\vec{r} \right] \right\}$$

$$T_{pq} = T_{ij} = \frac{q_j}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] (\vec{p}_i \times \vec{r})$$

$$T_{qp} = T_{ji} = -\frac{q_i}{r^3} \left[1 - 3\left(\frac{r}{r_c}\right)^{2} + 2\left(\frac{r}{r_c}\right)^{3} \right] (\vec{p}_j \times \vec{r})$$

$$T_{pp} = T_{ij} = -\frac{1}{r^3} \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + e3\left(\frac{r}{r_c}\right)^{4} \right] (\vec{p}_i \times \vec{p}_j) + \frac{3}{r^5} \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + 3\left(\frac{r}{r_c}\right)^{4} \right] (\vec{p}_j \bullet \vec{r})(\vec{p}_i \times \vec{r})$$

$$T_{pp} = T_{ji} = -\frac{1}{r^3} \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + 3\left(\frac{r}{r_c}\right)^{4} \right] (\vec{p}_j \times \vec{p}_i) + \frac{3}{r^5} \left[1 - 4\left(\frac{r}{r_c}\right)^{3} + 3\left(\frac{r}{r_c}\right)^{4} \right] (\vec{p}_i \bullet \vec{r})(\vec{p}_j \times \vec{r})$$

where ε and σ are the standard LJ parameters, r_c is the cutoff, q_i and q_j are the charges on the two particles, $\vec{p}_i$ and $\vec{p}_j$ are the dipole moment vectors of the two particles, r is their separation distance, and the vector r = Ri - Rj is the

separation vector between the two particles. Note that Eqq and Fqq are simply Coulombic energy and force, Fij = -Fji as symmetric forces, and Tij != -Tji since the torques do not act symmetrically. The shifted-force formula for the Lennard-Jones potential is reported in *(Stoddard)*. The original (non-shifted) formulas for the electrostatic potentials, forces and torques can be found in *(Price)*. The shifted-force electrostatic potentials have been obtained by applying equation 5.13 of *(AllenTildesley)*. The formulas for the corresponding forces and torques have been obtained by applying the 'chain rule' as in appendix C.3 of *(AllenTildesley)*.

If one cutoff is specified in the pair_style command, it is used for both the LJ and Coulombic (q,p) terms. If two cutoffs are specified, they are used as cutoffs for the LJ and Coulombic (q,p) terms respectively. This pair style also supports an optional *scale* keyword as part of a pair_coeff statement, where the interactions can be scaled according to this factor. This scale factor is also made available for use with fix adapt.

Style *lj/cut/dipole/long* computes the short-range portion of point-dipole interactions as discussed in *(Toukmaji)*. Dipole-dipole, dipole-charge, and charge-charge interactions are all supported, along with the standard 12/6 Lennard-Jones interactions, which are computed with a cutoff. A *kspace_style* must be defined to use this pair style. If only dipoles (not point charges) are included in the model, the kspace style can be one of these 3 options, all of which compute the long-range portion of dipole-dipole interactions. If the model includes point charges (in addition to dipoles), then only the first of these kspace styles can be used:

- *kspace_style ewald/disp*
- *kspace_style ewald/dipole*
- *kspace_style pppm/dipole*

Style *lj/long/dipole/long* has the same functionality as style *lj/cut/dipole/long*, except it also has an option to compute 12/6 Lennard-Jones interactions for use with a long-range dispersion kspace style. This is done by setting its *flag_lj* argument to *long*. For long-range LJ interactions, the *kspace_style ewald/disp* command must be used.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff1 (distance units)
- cutoff2 (distance units)

The latter 2 coefficients are optional. If not specified, the global LJ and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both LJ and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the LJ and Coulombic cutoffs for this type pair. When using a long-rang Coulomb solver, only a global Coulomb cutoff may be used and only the LJ cutoff may be changed with the *pair_coeff* command. When using the *lj/long/dipole/long* pair style with *long long* setting, only a single global cutoff may be provided and no cutoff for the *pair_coeff* command.

Note that for systems using these pair styles, typically particles should be able to exert torque on each other via their dipole moments so that the particle and its dipole moment can rotate. This requires they not be point particles, but finite-size spheres. Thus you should use a command like *atom_style hybrid sphere dipole* to use particles with both attributes.

The magnitude and orientation of the dipole moment for each particle can be defined by the *set* command or in the "Atoms" section of the data file read in by the *read_data* command.

Rotating finite-size particles have 6 degrees of freedom (DOFs), translation and rotational. You can use the *compute temp/sphere* command to monitor a temperature which includes all these DOFs.

Finite-size particles with dipole moments should be integrated using one of these options:

- *fix nve/sphere update dipole*
- *fix nve/sphere update dipole* plus *fix langevin omega yes*
- *fix nvt/sphere update dipole*
- *fix npt/sphere update dipole*

In all cases the "update dipole" setting ensures the dipole moments are also rotated when the finite-size spheres rotate. The 2nd and 3rd bullets perform thermostatting; in the case of a Langevin thermostat the "omega yes" option also thermostats the rotational degrees of freedom (if desired). The 4th bullet performs thermostatting and barostatting.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.80.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distances for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

For atom type pairs I,J and I != J, the A, sigma, d1, and d2 coefficients and cutoff distance for this pair style can be mixed. A is an energy value mixed like a LJ epsilon. D1 and d2 are distance values and are mixed like sigma. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction; such energy goes to zero at the cutoff by construction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.80.5 Restrictions

The *lj/cut/dipole/cut*, *lj/cut/dipole/long*, *lj/long/dipole/long*, and lj/sf/dipole/sf* styles are part of the DIPOLE package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Using dipole pair styles with *electron units* is not currently supported.

4.80.6 Related commands

pair_coeff , *set*, *read_data*, *fix nve/sphere*, *fix nvt/sphere*

4.80.7 Default

none

(AllenTildesley) Allen and Tildesley, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987.

(Toukmaji) Toukmaji, Sagui, Board, and Darden, J Chem Phys, 113, 10913 (2000).

(Stoddard) Stoddard and Ford, Phys Rev A, 8, 1504 (1973).

(Price) Price, Stone and Alderton, Mol Phys, 52, 987 (1984).

4.81 pair_style dpd command

Accelerator Variants: *dpd/gpu*, *dpd/intel*, *dpd/kk*, *dpd/omp*

4.82 pair_style dpd/tstat command

Accelerator Variants: *dpd/tstat/gpu*, *dpd/tstat/kk*, *dpd/tstat/omp*

4.82.1 Syntax

```
pair_style dpd T cutoff seed
pair_style dpd/tstat Tstart Tstop cutoff seed
```

- T = temperature (temperature units) (dpd only)
- Tstart,Tstop = desired temperature at start/end of run (temperature units) (dpd/tstat only)
- cutoff = global cutoff for DPD interactions (distance units)
- seed = random # seed (positive integer)

4.82.2 Examples

```
pair_style dpd 1.0 2.5 34387
pair_coeff * * 3.0 1.0
pair_coeff 1 1 3.0 1.0 1.0

pair_style hybrid/overlay lj/cut 2.5 dpd/tstat 1.0 1.0 2.5 34387
pair_coeff * * lj/cut 1.0 1.0
pair_coeff * * dpd/tstat 1.0
```

4.82.3 Description

Style *dpd* computes a force field for dissipative particle dynamics (DPD) following the exposition in *(Groot)*.

Style *dpd/tstat* invokes a DPD thermostat on pairwise interactions, which is equivalent to the non-conservative portion of the DPD force field. This pairwise thermostat can be used in conjunction with any *pair style*, and instead of per-particle thermostats like *fix langevin* or ensemble thermostats like Nose Hoover as implemented by *fix nvt*. To use *dpd/tstat* as a thermostat for another pair style, use the *pair_style hybrid/overlay* command to compute both the desired pair interaction and the thermostat for each pair of particles.

For style *dpd*, the force on atom I due to atom J is given as a sum of 3 terms

$$
\begin{aligned}
\vec{f} &= (F^C + F^D + F^R)\hat{r_{ij}} \qquad\qquad r < r_c \\
F^C &= A w(r) \\
F^D &= -\gamma w^2(r)(\hat{r_{ij}} \bullet \vec{v}_{ij}) \\
F^R &= \sigma w(r) \alpha (\Delta t)^{-1/2} \\
w(r) &= 1 - \frac{r}{r_c}
\end{aligned}
$$

where F^C is a conservative force, F^D is a dissipative force, and F^R is a random force. $\hat{r_{ij}}$ is a unit vector in the direction $r_i - r_j$, $\vec{v}_{ij}$ is the vector difference in velocities of the two atoms $\vec{v}_i - \vec{v}_j$, α is a Gaussian random number with zero mean and unit variance, dt is the timestep size, and $w(r)$ is a weighting factor that varies between 0 and 1. r_c is the pairwise cutoff. σ is set equal to $\sqrt{2 k_B T \gamma}$, where k_B is the Boltzmann constant and T is the temperature parameter in the pair_style command.

For style *dpd/tstat*, the force on atom I due to atom J is the same as the above equation, except that the conservative F^C term is dropped. Also, during the run, T is set each timestep to a ramped value from *Tstart* to *Tstop*.

For style *dpd*, the pairwise energy associated with style *dpd* is only due to the conservative force term F^C, and is shifted to be zero at the cutoff distance r_c. The pairwise virial is calculated using all 3 terms. For style *dpd/tstat* there is no pairwise energy, but the last two terms of the formula make a contribution to the virial.

For style *dpd*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (force units)
- γ (force/velocity units)
- cutoff (distance units)

The cutoff coefficient is optional. If not specified, the global DPD cutoff is used. Note that sigma is set equal to sqrt(2 T gamma), where T is the temperature set by the *pair_style* command so it does not need to be specified.

For style *dpd/tstat*, the coefficients defined for each pair of atoms types via the *pair_coeff* command are:

- γ (force/velocity units)

- cutoff (distance units)

The cutoff coefficient is optional.

Styles with a *gpu* suffix are implemented based on the work of *(Afshar)* and *(Phillips)*.

Note

If you are modeling DPD polymer chains, you may want to use the *pair_style srp* command in conjunction with these pair styles. It is a soft segmental repulsive potential (SRP) that can prevent DPD polymer chains from crossing each other.

Note

The virial calculation for pressure when using these pair styles includes all the components of force listed above, including the random force. Since the random force depends on random numbers, everything that changes the order of atoms in the neighbor list (e.g. different number of MPI ranks or a different neighbor list skin distance) will also change the sequence in which the random numbers are applied and thus the individual forces and therefore also the virial/pressure.

Note

For more consistent time integration and force computation you may consider using *fix mvv/dpd* instead of *fix nve*.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.82.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

These pair styles do not support the *pair_modify* shift option for the energy of the pair interaction. Note that as discussed above, the energy due to the conservative F^C term is already shifted to be 0.0 at the cutoff distance r_c.

The *pair_modify* table option is not relevant for these pair styles.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file. Note that the user-specified random number seed is stored in the restart file, so when a simulation is restarted, each processor will re-initialize its random number generator the same

way it did initially. This means the random forces will be random, but will not be the same as they would have been if the original simulation had continued past the restart time.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

The *dpd/tstat* style can ramp its target temperature over multiple runs, using the *start* and *stop* keywords of the *run* command. See the *run* command for details of how to do this.

4.82.5 Restrictions

These styles are part of the DPD-BASIC package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The default frequency for rebuilding neighbor lists is every 10 steps (see the *neigh_modify* command). This may be too infrequent for style *dpd* simulations since particles move rapidly and can overlap by large amounts. If this setting yields a non-zero number of "dangerous" reneighborings (printed at the end of a simulation), you should experiment with forcing reneighboring more often and see if system energies/trajectories change.

These pair styles requires you to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

These pair styles will not restart exactly when using the *read_restart* command, though they should provide statistically similar results. This is because the forces they compute depend on atom velocities. See the *read_restart* command for more details.

4.82.6 Related commands

pair_style dpd/ext, *pair_coeff*, *fix nvt*, *fix langevin*, *pair_style srp*, *fix mvv/dpd*.

4.82.7 Default

none

(Groot) Groot and Warren, J Chem Phys, 107, 4423-35 (1997).

(Afshar) Afshar, F. Schmid, A. Pishevar, S. Worley, Comput Phys Comm, 184, 1119-1128 (2013).

(Phillips) C. L. Phillips, J. A. Anderson, S. C. Glotzer, Comput Phys Comm, 230, 7191-7201 (2011).

4.83 pair_style dpd/coul/slater/long command

Accelerator Variants: *dpd/coul/slater/long/gpu*

4.83.1 Syntax

```
pair_style dpd/coul/slater/long T cutoff_DPD seed lambda cutoff_coul
```

- T = temperature (temperature units)
- cutoff_DPD = global cutoff for DPD interactions (distance units)
- seed = random # seed (positive integer)
- lambda = decay length of the charge (distance units)
- cutoff_coul = global cutoff for Coulombic interactions (distance units)

4.83.2 Examples

```
pair_style dpd/coul/slater/long 1.0 2.5 34387 0.25 3.0

pair_coeff 1 1 78.0 4.5          # not charged by default
pair_coeff 2 2 78.0 4.5 yes
```

4.83.3 Description

Added in version 27June2024.

Style *dpd/coul/slater/long* computes a force field for dissipative particle dynamics (DPD) following the exposition in *(Groot)*. It also allows for the use of charged particles in the model by adding a long-range Coulombic term to the DPD interactions. The short-range portion of the Coulombics is calculated by this pair style. The long-range Coulombics are computed by use of the *kspace_style* command, e.g. using the Ewald or PPPM styles.

Coulombic forces in mesoscopic models such as DPD employ potentials without explicit excluded-volume interactions. The goal is to prevent artificial ionic pair formation by including a charge distribution in the Coulomb potential, following the formulation in *(Melchor1)*.

Note

This pair style is effectively the combination of the *pair_style dpd* and *pair_style coul/slater/long* commands, but should be more efficient (especially on GPUs) than using *pair_style hybrid/overlay dpd coul/slater/long*. That is particularly true for the GPU package version of the pair style since this version is compatible with computing neighbor lists on the GPU instead of the CPU as is required for hybrid styles.

In the charged DPD model, the force on bead I due to bead J is given as a sum of 4 terms:

$$
\begin{aligned}
\vec{f} &= (F^C + F^D + F^R + F^E)\hat{r_{ij}} \\
F^C &= Aw(r) && r < r_{DPD} \\
F^D &= -\gamma w^2(r)(\hat{r_{ij}} \bullet \vec{v}_{ij}) && r < r_{DPD} \\
F^R &= \sigma w(r)\alpha(\Delta t)^{-1/2} && r < r_{DPD} \\
w(r) &= 1 - \frac{r}{r_{DPD}} \\
F^E &= \frac{Cq_iq_j}{\varepsilon r^2}\left(1 - exp\left(\frac{2r_{ij}}{\lambda}\right)\left(1 + \frac{2r_{ij}}{\lambda}\left(1 + \frac{r_{ij}}{\lambda}\right)\right)\right)
\end{aligned}
$$

where F^C is a conservative force, F^D is a dissipative force, F^R is a random force, and F^E is an electrostatic force. $\hat{r_{ij}}$ is a unit vector in the direction $r_i - r_j$, $\vec{v}_{ij}$ is the vector difference in velocities of the two atoms $\vec{v}_i - \vec{v}_j$, α is a Gaussian random number with zero mean and unit variance, dt is the timestep size, and $w(r)$ is a weighting factor that varies between 0 and 1.

σ is set equal to $\sqrt{2k_BT\gamma}$, where k_B is the Boltzmann constant and T is the temperature parameter in the pair_style command.

r_{DPD} is the pairwise cutoff for the first 3 DPD terms in the formula as specified by *cutoff_DPD*. For the F^E term, pairwise interactions within the specified *cutoff_coul* distance are computed directly; interactions beyond that distance are computed in reciprocal space. C is the same Coulomb conversion factor used in the Coulombic formulas described on the *pair_coul* doc page.

The following parameters must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (force units)
- γ (force/velocity units)
- is_charged (optional boolean, default = no)

The *is_charged* parameter is optional and can be specified as *yes* or *no*. *Yes* should be used for interactions between two types of charged particles. *No* is the default and should be used for interactions between two types of particles when one or both are uncharged.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.83.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file. Note that the user-specified random number seed is stored in the restart file, so when a simulation is restarted, each processor will re-initialize its random number generator the same way it did initially. This means the random forces will be random, but will not be the same as they would have been if the original simulation had continued past the restart time.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.83.5 Restrictions

This style is part of the DPD-BASIC package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The default frequency for rebuilding neighbor lists is every 10 steps (see the *neigh_modify* command). This may be too infrequent since particles move rapidly and can overlap by large amounts. If this setting yields a non-zero number of "dangerous" reneighborings (printed at the end of a simulation), you should experiment with forcing reneighboring more often and see if system energies/trajectories change.

This pair style requires use of the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

This pair style also requires use of a long-range solvers from the KSPACE package.

This pair style will not restart exactly when using the *read_restart* command, though they should provide statistically similar results. This is because the forces they compute depend on atom velocities. See the *read_restart* command for more details.

4.83.6 Related commands

pair_style dpd, *pair_style coul/slater/long*,

4.83.7 Default

For the pair_coeff command, the default is is_charged = no.

(Groot) Groot and Warren, J Chem Phys, 107, 4423-35 (1997).

(Melchor) Gonzalez-Melchor, Mayoral, Velazquez, and Alejandre, J Chem Phys, 125, 224107 (2006).

4.84 pair_style dpd/ext command

Accelerator Variants: dpd/ext/kk dpd/ext/omp

4.85 pair_style dpd/ext/tstat command

Accelerator Variants: dpd/ext/tstat/kk dpd/ext/tstat/omp

4.85.1 Syntax

```
pair_style dpd/ext T cutoff seed
pair_style dpd/ext/tstat Tstart Tstop cutoff seed
```

- T = temperature (temperature units)
- Tstart,Tstop = desired temperature at start/end of run (temperature units)
- cutoff = global cutoff for DPD interactions (distance units)
- seed = random # seed (positive integer)

4.85.2 Examples

```
pair_style dpd/ext 1.0 2.5 34387
pair_coeff 1 1 25.0 4.5 4.5 0.5 0.5 1.2
pair_coeff 1 2 40.0 4.5 4.5 0.5 0.5 1.2

pair_style hybrid/overlay lj/cut 2.5 dpd/ext/tstat 1.0 1.0 2.5 34387
pair_coeff * * lj/cut 1.0 1.0
pair_coeff * * 4.5 4.5 0.5 0.5 1.2
```

4.85.3 Description

The style *dpd/ext* computes an extended force field for dissipative particle dynamics (DPD) following the exposition in *(Groot)*, *(Junghans)*.

Style *dpd/ext/tstat* invokes an extended DPD thermostat on pairwise interactions, equivalent to the non-conservative portion of the extended DPD force field. To use *dpd/ext/tstat* as a thermostat for another pair style, use the *pair_style hybrid/overlay* command to compute both the desired pair interaction and the thermostat for each pair of particles.

For the style *dpd/ext*, the force on atom I due to atom J is given as a sum of 3 terms

$$
\begin{aligned}
\mathbf{f} &= f^C + f^D + f^R \qquad\qquad r < r_c \\
f^C &= A_{ij} w(r) \hat{\mathbf{r}}_{ij} \\
f^D &= -\gamma_{\parallel} w_{\parallel}^2(r)(\hat{\mathbf{r}}_{ij} \cdot \mathbf{v}_{ij})\hat{\mathbf{r}}_{ij} - \gamma_{\perp} w_{\perp}^2(r)(\mathbf{I} - \hat{\mathbf{r}}_{ij}\hat{\mathbf{r}}_{ij}^{\mathrm{T}})\mathbf{v}_{ij} \\
f^R &= \sigma_{\parallel} w_{\parallel}(r) \frac{\alpha}{\sqrt{\Delta t}} \hat{\mathbf{r}}_{ij} + \sigma_{\perp} w_{\perp}(r)(\mathbf{I} - \hat{\mathbf{r}}_{ij}\hat{\mathbf{r}}_{ij}^{\mathrm{T}}) \frac{\xi_{ij}}{\sqrt{\Delta t}} \\
w(r) &= 1 - r/r_c
\end{aligned}
$$

where $\mathbf{f}^C$ is a conservative force, $\mathbf{f}^D$ is a dissipative force, and $\mathbf{f}^R$ is a random force. A_{ij} is the maximum repulsion between the two atoms, $\hat{\mathbf{r}}_{ij}$ is a unit vector in the direction $\mathbf{r}_i - \mathbf{r}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the vector difference in velocities of the two atoms, α and ξ_{ij} are Gaussian random numbers with zero mean and unit variance, Δt is the timestep, $w(r) = 1 - r/r_c$ is a weight function for the conservative interactions that varies between 0 and 1, r_c is the corresponding cutoff, $w_\alpha(r) = (1 - r/\bar{r}_c)^{s_\alpha}$, $\alpha \equiv (\parallel, \perp)$, are weight functions with coefficients s_α that vary between 0 and 1, $\bar{r}_c$ is the corresponding cutoff, $\mathbf{I}$ is the unit matrix, $\sigma_\alpha = \sqrt{2 k_B T \gamma_\alpha}$, where k_B is the Boltzmann constant and T is the temperature in the pair_style command.

For the style *dpd/ext/tstat*, the force on atom I due to atom J is the same as the above equation, except that the conservative $\mathbf{f}^C$ term is dropped. Also, during the run, T is set each timestep to a ramped value from Tstart to Tstop.

For the style *dpd/ext*, the pairwise energy associated with style *dpd/ext* is only due to the conservative force term $\mathbf{f}^C$, and is shifted to be zero at the cutoff distance r_c. The pairwise virial is calculated using all three terms. There is no pairwise energy for style *dpd/ext/tstat*, but the last two terms of the formula contribute the virial.

For the style *dpd/ext/tstat*, the force on atom I due to atom J is the same as the above equation, except that the conservative $\mathbf{f}^C$ term is dropped. Also, during the run, T is set each timestep to a ramped value from Tstart to Tstop.

For the style *dpd/ext*, the pairwise energy associated with style *dpd/ext* is only due to the conservative force term $\mathbf{f}^C$, and is shifted to be zero at the cutoff distance r_c. The pairwise virial is calculated using all three terms. There is no pairwise energy for style *dpd/ext/tstat*, but the last two terms of the formula contribute the virial.

For the style *dpd/ext*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above:

- A (force units)
- $\gamma_{\parallel}$ (force/velocity units)

- $\gamma_\perp$ (force/velocity units)
- $s_\parallel$ (unitless)
- $s_\perp$ (unitless)
- r_c (distance units)

The last coefficient is optional. If not specified, the global DPD cutoff is used. Note that σ's are set equal to $\sqrt{2k_BT\gamma}$, where T is the temperature set by the *pair_style* command so it does not need to be specified.

For the style *dpd/ext/tstat*, the coefficients defined for each pair of atoms types via the *pair_coeff* command are:

- $\gamma_\parallel$ (force/velocity units)
- $\gamma_\perp$ (force/velocity units)
- $s_\parallel$ (unitless)
- $s_\perp$ (unitless)
- r_c (distance units)

The last coefficient is optional.

Note

If you are modeling DPD polymer chains, you may want to use the *pair_style srp* command in conjunction with these pair styles. It is a soft segmental repulsive potential (SRP) that can prevent DPD polymer chains from crossing each other.

Note

The virial calculation for pressure when using these pair styles includes all the components of force listed above, including the random force. Since the random force depends on random numbers, everything that changes the order of atoms in the neighbor list (e.g. different number of MPI ranks or a different neighbor list skin distance) will also change the sequence in which the random numbers are applied and thus the individual forces and therefore also the virial/pressure.

Note

For more consistent time integration and force computation you may consider using *fix mvv/dpd* instead of *fix nve*.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

Mixing, shift, table, tail correction, restart, rRESPA info:

The style *dpd/ext* does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The pair styles do not support the *pair_modify* shift option for the energy of the pair interaction. Note that as discussed above, the energy due to the conservative $\mathbf{f}^C$ term is already shifted to be zero at the cutoff distance r_c.

The *pair_modify* table option is not relevant for the style *dpd/ext*.

The style *dpd/ext* does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

The pair styles can only be used via the pair keyword of the *run_style respa* command. They do not support the *inner*, *middle*, and *outer*keywords.

The style *dpd/ext/tstat* can ramp its target temperature over multiple runs, using the start and stop keywords of the *run* command. See the *run* command for details of how to do this.

4.85.4 Restrictions

These styles are part of the DPD-BASIC package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The default frequency for rebuilding neighbor lists is every 10 steps (see the *neigh_modify* command). This may be too infrequent for style *dpd/ext* simulations since particles move rapidly and can overlap by large amounts. If this setting yields a non-zero number of say{dangerous} reneighborings (printed at the end of a simulation), you should experiment with forcing reneighboring more often and see if system energies/trajectories change.

The pair styles require to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

The pair styles will not restart exactly when using the *read_restart* command, though they should provide statistically similar results. This is because the forces they compute depend on atom velocities. See the *read_restart* command for more details.

4.85.5 Related commands

pair_style dpd, *pair_coeff* , *fix nvt*, *fix langevin*, *pair_style srp*, *fix mvv/dpd*.

Default: none

(Groot) Groot and Warren, J Chem Phys, 107, 4423-35 (1997).

(Junghans) Junghans, Praprotnik and Kremer, Soft Matter 4, 156, 1119-1128 (2008).

4.86 pair_style dpd/fdt command

4.87 pair_style dpd/fdt/energy command

Accelerator Variants: *dpd/fdt/energy/kk*

4.87.1 Syntax

```
pair_style style args
```

- style = *dpd/fdt* or *dpd/fdt/energy*
- args = list of arguments for a particular style

dpd/fdt args = T cutoff seed
  T = temperature (temperature units)
  cutoff = global cutoff for DPD interactions (distance units)
  seed = random # seed (positive integer)
dpd/fdt/energy args = cutoff seed
  cutoff = global cutoff for DPD interactions (distance units)
  seed = random # seed (positive integer)

4.87.2 Examples

```
pair_style dpd/fdt 300.0 2.5 34387
pair_coeff * * 3.0 1.0 2.5

pair_style dpd/fdt/energy 2.5 34387
pair_coeff * * 3.0 1.0 0.1 2.5
```

4.87.3 Description

Styles *dpd/fdt* and *dpd/fdt/energy* compute the force for dissipative particle dynamics (DPD) simulations. The *dpd/fdt* style is used to perform DPD simulations under isothermal and isobaric conditions, while the *dpd/fdt/energy* style is used to perform DPD simulations under isoenergetic and isoenthalpic conditions (see *(Lisal)*). For DPD simulations in general, the force on atom I due to atom J is given as a sum of 3 terms

$$\begin{aligned} \vec{f} &= (F^C + F^D + F^R)\hat{r_{ij}} \qquad r < r_c \\ F^C &= Aw(r) \\ F^D &= -\gamma w^2(r)(\hat{r_{ij}} \bullet \vec{v}_{ij}) \\ F^R &= \sigma w(r)\alpha(\Delta t)^{-1/2} \\ w(r) &= 1 - \frac{r}{r_c} \end{aligned}$$

where F^C is a conservative force, F^D is a dissipative force, and F^R is a random force. $\hat{r_{ij}}$ is a unit vector in the direction $r_i - r_j$, $\vec{v}_{ij}$ is the vector difference in velocities of the two atoms, $\vec{v}_i - \vec{v}_j$, α is a Gaussian random number with zero mean and unit variance, dt is the timestep size, and $w(r)$ is a weighting factor that varies between 0 and 1, r_c is the pairwise cutoff. Note that alternative definitions of the weighting function exist, but would have to be implemented as a separate pair style command.

For style *dpd/fdt*, the fluctuation-dissipation theorem defines γ to be set equal to $\sigma^2/(2T)$, where T is the set point temperature specified as a pair style parameter in the above examples. The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (force units)
- σ (force*time^(1/2) units)

- cutoff (distance units)

The last coefficient is optional. If not specified, the global DPD cutoff is used.

Style *dpd/fdt/energy* is used to perform DPD simulations under isoenergetic and isoenthalpic conditions. The fluctuation-dissipation theorem defines γ to be set equal to $\sigma^2/(2\theta)$, where θ is the average internal temperature for the pair. The particle internal temperature is related to the particle internal energy through a mesoparticle equation of state (see *fix eos*). The differential internal conductive and mechanical energies are computed within style *dpd/fdt/energy* as:

$$du_i^{cond} = \kappa_{ij}(\frac{1}{\theta_i} - \frac{1}{\theta_j})\omega_{ij}^2 + \alpha_{ij}\omega_{ij}\zeta_{ij}^q(\Delta t)^{-1/2}$$

$$du_i^{mech} = -\frac{1}{2}\gamma_{ij}\omega_{ij}^2(\frac{\vec{r}_{ij}}{r_{ij}} \bullet \vec{v}_{ij})^2 - \frac{\sigma_{ij}^2}{4}(\frac{1}{m_i} + \frac{1}{m_j})\omega_{ij}^2 - \frac{1}{2}\sigma_{ij}\omega_{ij}(\frac{\vec{r}_{ij}}{r_{ij}} \bullet \vec{v}_{ij})\zeta_{ij}(\Delta t)^{-1/2}$$

where

$$\alpha_{ij}^2 = 2k_B\kappa_{ij}$$

$$\sigma_{ij}^2 = 2\gamma_{ij}k_B\Theta_{ij}$$

$$\Theta_{ij}^{-1} = \frac{1}{2}(\frac{1}{\theta_i} + \frac{1}{\theta_j})$$

$\zeta_i j^q$ is a second Gaussian random number with zero mean and unit variance that is used to compute the internal conductive energy. The fluctuation-dissipation theorem defines $alpha^2$ to be set equal to $2k_B\kappa$, where κ is the mesoparticle thermal conductivity parameter. The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (force units)
- σ (force*time^(1/2) units)
- κ (energy*temperature/time units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global DPD cutoff is used.

The pairwise energy associated with styles *dpd/fdt* and *dpd/fdt/energy* is only due to the conservative force term F^C, and is shifted to be zero at the cutoff distance r_c. The pairwise virial is calculated using only the conservative term.

The forces computed through the *dpd/fdt* and *dpd/fdt/energy* styles can be integrated with the velocity-Verlet integration scheme or the Shardlow splitting integration scheme described by *(Lisal)*. In the cases when these pair styles are combined with the *fix shardlow*, these pair styles differ from the other dpd styles in that the dissipative and random forces are split from the force calculation and are not computed within the pair style. Thus, only the conservative force is computed by the pair style, while the stochastic integration of the dissipative and random forces are handled through the Shardlow splitting algorithm approach. The Shardlow splitting algorithm is advantageous, especially when performing DPD under isoenergetic conditions, as it allows significantly larger timesteps to be taken.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.87.4 Restrictions

These commands are part of the DPD-REACT package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Pair styles *dpd/fdt* and *dpd/fdt/energy* require use of the *comm_modify vel yes* option so that velocities are stored by ghost atoms.

Pair style *dpd/fdt/energy* requires *atom_style dpd* to be used in order to properly account for the particle internal energies and temperatures.

4.87.5 Related commands

pair_coeff, *fix shardlow*

4.87.6 Default

none

(Lisal) M. Lisal, J.K. Brennan, J. Bonet Avalos, J. Chem. Phys., 135, 204105 (2011).

4.88 pair_style drip command

4.88.1 Syntax

```
pair_style hybrid/overlay drip [styles ...]
```

- styles = other styles to be overlayed with drip (optional)

4.88.2 Examples

```
pair_style hybrid/overlay drip
pair_coeff * * none
pair_coeff * * drip  C.drip  C

pair_style hybrid/overlay drip rebo
pair_coeff * * drip  C.drip     C
pair_coeff * * rebo  CH.airebo  C

pair_style hybrid/overlay drip rebo
pair_coeff * * drip  C.drip     C NULL
pair_coeff * * rebo  CH.airebo  C H
```

4.88.3 Description

Style *drip* computes the interlayer interactions of layered materials using the dihedral-angle-corrected registry-dependent (DRIP) potential as described in *(Wen)*, which is based on the *(Kolmogorov)* potential and provides an improved prediction for forces. The total potential energy of a system is

$$E = \frac{1}{2} \sum_i \sum_{j \notin \text{layer } i} \phi_{ij}$$

$$\phi_{ij} = f_c(x_r) \left[e^{-\lambda(r_{ij}-z_0)} \left[C + f(\rho_{ij}) + g(\rho_{ij}, \{\alpha_{ij}^{(m)}\}) \right] - A \left(\frac{z_0}{r_{ij}} \right)^6 \right]$$

where the r^{-6} term models the attractive London dispersion, the exponential term is designed to capture the registry effect due to overlapping *pi* bonds, and *fc* is a cutoff function.

This potential (DRIP) only provides the interlayer interactions between graphene layers. So, to perform a realistic simulation, it should be used in combination with an intralayer potential such as *REBO* and *Tersoff*. To keep the intralayer interactions unaffected, we should avoid applying DRIP to contribute energy to intralayer interactions. This can be achieved by assigning different molecular IDs to atoms in different layers, and DRIP is implemented such that only atoms with different molecular ID can interact with each other. For this purpose, *atom style* "molecular" or "full" has to be used.

On the other way around, *REBO* (*Tersoff* or any other potential used to provide the intralayer interactions) should not interfere with the interlayer interactions described by DRIP. This is typically automatically achieved using the commands provided in the *Examples* section above, since the cutoff distance for carbon-carbon interaction in the intralayer potentials (e.g. 2 Angstrom for *REBO*) is much smaller than the equilibrium layer distance of graphene layers (about 3.4 Angstrom). If you want, you can enforce this by assigning different atom types to atoms in different layers, and apply an intralayer potential to one atom type. See *pair_hybrid* for details.

The *pair_coeff* command for DRIP takes *4+N* arguments, where *N* is the number of LAMMPS atom types. The fist three arguments must be fixed to be * * *drip*, the fourth argument is the path to the DRIP parameter file, and the remaining N arguments specifying the mapping between element in the parameter file and atom types. For example, if your LAMMPS simulation has 3 atom types and you want all of them to be C, you would use the following pair_coeff command:

```
pair_coeff * * drip  C.drip  C C C
```

If a mapping value is specified as NULL, the mapping is not performed. This could be useful when DRIP is used to model part of the system where other element exists. Suppose you have a hydrocarbon system, with C of atom type 1 and H of atom type 2, you can use the following command to inform DRIP not to model H atoms:

```
pair_style hybrid/overlay drip rebo
pair_coeff * * drip  C.drip     C NULL
pair_coeff * * rebo  CH.airebo  C H
```

Note

The potential parameters developed in *(Wen)* are provided with LAMMPS (see the "potentials" directory). Besides those in *Wen*, an additional parameter "normal_cutoff", specific to the LAMMPS implementation, is used to find the three nearest neighbors of an atom to construct the normal.

4.88.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.88.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The *C.drip* parameter file provided with LAMMPS (see the "potentials" directory) is parameterized for metal *units*. You can use the DRIP potential with any LAMMPS units, but you would need to create your own custom parameter file with coefficients listed in the appropriate units, if your simulation does not use "metal" units.

4.88.6 Related commands

pair_style lebedeva_z, *pair_style kolmogorov/crespi/z*, *pair_style kolmogorov/crespi/full*, *pair_style ilp/graphene/hbn*.

(Wen) M. Wen, S. Carr, S. Fang, E. Kaxiras, and E. B. Tadmor, Phys. Rev. B, 98, 235404 (2018)

(Kolmogorov) A. N. Kolmogorov, V. H. Crespi, Phys. Rev. B 71, 235415 (2005)

4.89 pair_style dsmc command

4.89.1 Syntax

```
pair_style dsmc max_cell_size seed weighting Tref Nrecompute Nsample
```

- max_cell_size = global maximum cell size for DSMC interactions (distance units)
- seed = random # seed (positive integer)
- weighting = macroparticle weighting
- Tref = reference temperature (temperature units)
- Nrecompute = re-compute v*sigma_max every this many timesteps (timesteps)
- Nsample = sample this many times in recomputing v*sigma_max

4.89.2 Examples

```
pair_style dsmc 2.5 34387 10 1.0 100 20
pair_coeff * * 1.0
pair_coeff 1 1 1.0
```

4.89.3 Description

Style *dsmc* computes collisions between pairs of particles for a direct simulation Monte Carlo (DSMC) model following the exposition in *(Bird)*. Each collision resets the velocities of the two particles involved. The number of pairwise collisions for each pair or particle types and the length scale within which they occur are determined by the parameters of the pair_style and pair_coeff commands.

Stochastic collisions are performed using the variable hard sphere (VHS) approach, with the user-defined *max_cell_size* value used as the maximum DSMC cell size, and reference cross-sections for collisions given using the pair_coeff command.

There is no pairwise energy or virial contributions associated with this pair style.

The following coefficient must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- sigma (area units, i.e. distance-squared)

The global DSMC *max_cell_size* determines the maximum cell length used in the DSMC calculation. A structured mesh is overlayed on the simulation box such that an integer number of cells are created in each direction for each processor's subdomain. Cell lengths are adjusted up to the user-specified maximum cell size.

To perform a DSMC simulation with LAMMPS, several additional options should be set in your input script, though LAMMPS does not check for these settings.

Since this pair style does not compute particle forces, you should use the "fix nve/noforce" time integration fix for the DSMC particles, e.g.

```
fix 1 all nve/noforce
```

This pair style assumes that all particles will communicated to neighboring processors every timestep as they move. This makes it possible to perform all collisions between pairs of particles that are on the same processor. To ensure this occurs, you should use these commands:

```
neighbor 0.0 bin
neigh_modify every 1 delay 0 check no
atom_modify sort 0 0.0
communicate single cutoff 0.0
```

These commands ensure that LAMMPS communicates particles to neighboring processors every timestep and that no ghost atoms are created. The output statistics for a simulation run should indicate there are no ghost particles or neighbors.

In order to get correct DSMC collision statistics, users should specify a Gaussian velocity distribution when populating the simulation domain. Note that the default velocity distribution is uniform, which will not give good DSMC collision rates. Specify "dist gaussian" when using the *velocity* command as in the following:

```
velocity all create 594.6 87287 loop geom dist gaussian
```

4.89.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file. Note that the user-specified random number seed is stored in the restart file, so when a simulation is restarted, each processor will re-initialize its random number generator the same way it did initially. This means the random forces will be random, but will not be the same as they would have been if the original simulation had continued past the restart time.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.89.5 Restrictions

This pair style is part of the MC package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires an *atom style* with per atom type masses.

4.89.6 Related commands

pair_coeff, *fix nve/noforce*, *neigh_modify*, *neighbor*, *comm_modify*

4.89.7 Default

none

(Bird) G. A. Bird, "Molecular Gas Dynamics and the Direct Simulation of Gas Flows" (1994).

4.90 pair_style e3b command

4.90.1 Syntax

```
pair_style e3b Otype
```

- Otype = atom type (numeric or type label) for oxygen

```
pair_coeff * * keyword
```

- one or more keyword/value pairs must be appended.
- keyword = *preset* or *Ea* or *Eb* or *Ec* or *E2* or *K3* or *K2* or *Rs* or *Rc3* or *Rc2* or *bondL* or *neigh*

- If the *preset* keyword is given, no others are needed. Otherwise, all are mandatory except for *neigh*. The *neigh* keyword is always optional.

preset arg = 2011 or 2015 = which set of predefined parameters to use
  2011 = use the potential parameters from (Tainter 2011)
  2015 = use the potential parameters from (Tainter 2015)
Ea arg = three-body energy for type A hydrogen bonding interactions (energy units)
Eb arg = three-body energy for type B hydrogen bonding interactions (energy units)
Ec arg = three-body energy for type C hydrogen bonding interactions (energy units)
E2 arg = two-body energy correction (energy units)
K3 arg = three-body exponential constant (inverse distance units)
K2 arg = two-body exponential constant (inverse distance units)
Rc3 arg = three-body cutoff (distance units)
Rc2 arg = two-body cutoff (distance units)
Rs arg = three-body switching function cutoff (distance units)
bondL arg = intramolecular OH bond length (distance units)
neigh arg = approximate integer number of molecules within Rc3 of an oxygen atom

4.90.2 Examples

```
pair_style e3b 1
pair_coeff * * Ea 35.85 Eb -240.2 Ec 449.3 E2 108269.9 K3 1.907 K2 4.872 Rc3 5.2 Rc2 5.2 Rs 5.0
↪bondL 0.9572

pair_style hybrid/overlay e3b 1 lj/cut/tip4p/long 1 2 1 1 0.15 8.5
pair_coeff * * e3b preset 2011

pair_style e3b OW
labelmap atom 1 C 2 H 3 O 4 N 5 OW 6 HW
pair_coeff * * Ea 35.85 Eb -240.2 Ec 449.3 E2 108269.9 K3 1.907 K2 4.872 Rc3 5.2 Rc2 5.2 Rs 5.0
↪bondL 0.9572
```

Used in example input script:

```
examples/PACKAGES/e3b/in.e3b-tip4p2005
```

4.90.3 Description

The *e3b* style computes an "explicit three-body" (E3B) potential for water *(Kumar 2008)*.

$$E = E_2 \sum_{i,j} e^{-k_2 r_{ij}} + E_A \sum_{\substack{i,j,k,\ell \\ \in \text{type A}}} f(r_{ij}) f(r_{k\ell}) + E_B \sum_{\substack{i,j,k,\ell \\ \in \text{type B}}} f(r_{ij}) f(r_{k\ell}) + E_C \sum_{\substack{i,j,k,\ell \\ \in \text{type C}}} f(r_{ij}) f(r_{k\ell})$$

$$f(r) = e^{-k_3 r} s(r)$$

$$s(r) = \begin{cases} 1 & r < R_s \\ \dfrac{(R_f - r)^2 (R_f - 3R_s + 2r)}{(R_f - R_s)^3} & R_s \le r \le R_f \\ 0 & r > R_f \end{cases}$$

This potential was developed as a water model that includes the three-body cooperativity of hydrogen bonding explicitly. To use it in this way, it must be applied in conjunction with a conventional two-body water model, through pair

style *hybrid/overlay*. The three body interactions are split into three types: A, B, and C. Type A corresponds to anti-cooperative double hydrogen bond donor interactions. Type B corresponds to the cooperative interaction of molecules that both donate and accept a hydrogen bond. Type C corresponds to anti-cooperative double hydrogen bond acceptor interactions. The three-body interactions are smoothly cutoff by the switching function s(r) between Rs and Rc3. The two-body interactions are designed to correct for the effective many-body interactions implicitly included in the conventional two-body potential. The two-body interactions are cut off sharply at Rc2, because K3 is typically significantly smaller than K2. See *(Kumar 2008)* for more details.

Only a single *pair_coeff* command is used with the *e3b* style and the first two arguments must be * *. The oxygen atom type for the pair style is passed as the only argument to the *pair_style* command, not in the *pair_coeff* command. The hydrogen atom type is inferred from the ordering of the atoms.

Note

Every atom of type Otype must be part of a water molecule. Each water molecule must have consecutive IDs with the oxygen first. This pair style does not test that this criteria is met.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

The *pair_coeff* command must have at least one keyword/value pair, as described above. The *preset* keyword sets the potential parameters to the values used in *(Tainter 2011)* or *(Tainter 2015)*. To use the water models defined in those references, the *e3b* style should always be used in conjunction with an *lj/cut/tip4p/long* style through *pair_style hybrid/overlay*, as demonstrated in the second example above. The *preset 2011* option should be used with the *TIP4P water model*. The *preset 2015* option should be used with the *TIP4P/2005 water model*. If the *preset* keyword is used, no other keyword is needed. Changes to the preset parameters can be made by specifying the *preset* keyword followed by the specific parameter to change, like *Ea*. Note that the other keywords must come after *preset* in the pair_style command. The *e3b* style can also be used to implement any three-body potential of the same form by specifying all the keywords except *neigh*: *Ea*, *Eb*, *Ec*, *E2*, *K3*, *K2*, *Rc3*, *Rc2*, *Rs*, and *bondL*. The keyword *bondL* specifies the intramolecular OH bond length of the water model being used. This is needed to include H atoms that are within the cutoff even when the attached oxygen atom is not.

This pair style allocates arrays sized according to the number of pairwise interactions within Rc3. To do this it needs an estimate for the number of water molecules within Rc3 of an oxygen atom. This estimate defaults to 10 and can be changed using the *neigh* keyword, which takes an integer as an argument. If the neigh setting is too small, the simulation will fail with the error "neigh is too small". If the neigh setting is too large, the pair style will use more memory than necessary.

This pair style tallies a breakdown of the total E3B potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 4. The 4 values correspond to the terms in the first equation above: the E2 term, the Ea term, the Eb term, and the Ec term.

See the examples/PACKAGES/e3b directory for a complete example script.

4.90.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style is incompatible with *respa*.

4.90.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

This pair style requires a fixed number of atoms in the simulation, so it is incompatible with fixes like *fix deposit*. If the number of atoms changes between runs, this pair style must be re-initialized by calling the *pair_style* and *pair_coeffs* commands. This is not a fundamental limitation of the pair style, but the code currently does not support a variable number of atoms.

The *preset* keyword currently only works with real, metal, si, and cgs *units*.

4.90.6 Related commands

pair_coeff, *compute pair*

4.90.7 Default

The option default for the *neigh* keyword is 10.

(Kumar) Kumar and Skinner, J. Phys. Chem. B, 112, 8311 (2008)

(Tainter 2011) Tainter, Pieniazek, Lin, and Skinner, J. Chem. Phys., 134, 184501 (2011)

(Tainter 2015) Tainter, Shi, and Skinner, 11, 2268 (2015)

4.91 pair_style eam command

Accelerator Variants: *eam/gpu*, *eam/intel*, *eam/kk*, *eam/omp*, *eam/opt*

4.92 pair_style eam/alloy command

Accelerator Variants: *eam/alloy/gpu*, *eam/alloy/intel*, *eam/alloy/kk*, *eam/alloy/omp*, *eam/alloy/opt*

4.93 pair_style eam/cd command

4.94 pair_style eam/cd/old command

4.95 pair_style eam/fs command

4.96 pair_style eam/he command

Accelerator Variants: *eam/fs/gpu*, *eam/fs/intel*, *eam/fs/kk*, *eam/fs/omp*, *eam/fs/opt*

4.96.1 Syntax

```
pair_style style
```

- style = *eam* or *eam/alloy* or *eam/cd* or *eam/cd/old* or *eam/fs* or *eam/he*

4.96.2 Examples

```
pair_style eam
pair_coeff * * cuu3
pair_coeff 1*3 1*3 niu3.eam

pair_style eam/alloy
pair_coeff * * ../potentials/NiAlH_jea.eam.alloy Ni Al Ni Ni

pair_style eam/cd
pair_coeff * * ../potentials/FeCr.cdeam Fe Cr

pair_style eam/fs
pair_coeff * * NiAlH_jea.eam.fs Ni Al Ni Ni

pair_style eam/he
pair_coeff * * PdHHe.eam.he Pd H He
```

4.96.3 Description

Style *eam* computes pairwise interactions for metals and metal alloys using embedded-atom method (EAM) potentials *(Daw)*. The total energy Ei of an atom I is given by

$$E_i = F_\alpha \left(\sum_{j \neq i} \rho_\beta(r_{ij}) \right) + \frac{1}{2} \sum_{j \neq i} \phi_{\alpha\beta}(r_{ij})$$

where F is the embedding energy which is a function of the atomic electron density rho, phi is a pair potential interaction, and alpha and beta are the element types of atoms I and J. The multi-body nature of the EAM potential is a result of the embedding energy term. Both summations in the formula are over all neighbors J of atom I within the cutoff distance.

The cutoff distance and the tabulated values of the functionals F, rho, and phi are listed in one or more files which are specified by the *pair_coeff* command. These are ASCII text files in a DYNAMO-style format which is described below. DYNAMO was the original serial EAM MD code, written by the EAM originators. Several DYNAMO potential files for different metals are included in the "potentials" directory of the LAMMPS distribution. All of these files are parameterized in terms of LAMMPS *metal units*.

Note

The *eam* style reads single-element EAM potentials in the DYNAMO *funcfl* format. Either single element or alloy systems can be modeled using multiple *funcfl* files and style *eam*. For the alloy case LAMMPS mixes the single-element potentials to produce alloy potentials, the same way that DYNAMO does. Alternatively, a single DYNAMO *setfl* file or Finnis/Sinclair EAM file can be used by LAMMPS to model alloy systems by invoking the *eam/alloy* or *eam/cd* or *eam/fs* or *eam/he* styles as described below. These files require no mixing since they specify alloy interactions explicitly.

Note

Note that unlike for other potentials, cutoffs for EAM potentials are not set in the pair_style or pair_coeff command; they are specified in the EAM potential files themselves. Likewise, valid EAM potential files usually contain atomic masses; thus you may not need to use the *mass* command to specify them, unless the potential file uses a dummy value (e.g. 0.0). LAMMPS will print a warning, if this is the case.

There are web sites that distribute and document EAM potentials stored in DYNAMO or other formats:

- https://www.ctcms.nist.gov/potentials
- https://openkim.org

These potentials should be usable with LAMMPS, though the alternate formats would need to be converted to the DYNAMO format used by LAMMPS and described on this page. The NIST site is maintained by Chandler Becker (cbecker at nist.gov) who is good resource for info on interatomic potentials and file formats.

The OpenKIM Project at https://openkim.org/browse/models/by-type provides EAM potentials that can be used directly in LAMMPS with the *kim command* interface.

Warning

The EAM potential files tabulate the embedding energy as a function of the local electron density ρ. When atoms get too close, this electron density may exceed the range for which the embedding energy was tabulated for. To avoid crashes, LAMMPS will assume a linearly increasing embedding energy for electron densities beyond the maximum tabulated value. LAMMPS will print a warning when this happens. It may be acceptable at the beginning of an

equilibration (e.g. when using randomized coordinates) but would be a big concern for accuracy if it happens during production runs. The EAM potential file triggering the warning during production is thus not a good choice, and the EAM model in general not likely a good model for the kind of system under investigation.

For style *eam*, potential values are read from a file that is in the DYNAMO single-element *funcfl* format. If the DYNAMO file was created by a Fortran program, it cannot have "D" values in it for exponents. C only recognizes "e" or "E" for scientific notation.

For style *eam* a potential file must be assigned to each I,I pair of atom types by using one or more pair_coeff commands, each with a single argument:

- filename

Thus the following command

```
pair_coeff *2 1*2 cuu3.eam
```

will read the cuu3 potential file and use the tabulated Cu values for F, phi, rho that it contains for type pairs 1,1 and 2,2 (type pairs 1,2 and 2,1 are ignored). See the *pair_coeff* doc page for alternate ways to specify the path for the potential file. In effect, this makes atom types 1 and 2 in LAMMPS be Cu atoms. Different single-element files can be assigned to different atom types to model an alloy system. The mixing to create alloy potentials for type pairs with I != J is done automatically the same way that the serial DYNAMO code originally did it; you do not need to specify coefficients for these type pairs.

Funcfl files in the *potentials* directory of the LAMMPS distribution have an ".eam" suffix. A DYNAMO single-element *funcfl* file is formatted as follows:

- line 1: comment (ignored)
- line 2: atomic number, mass, lattice constant, lattice type (e.g. FCC)
- line 3: Nrho, drho, Nr, dr, cutoff

On line 2, all values but the mass are ignored by LAMMPS. The mass is in mass *units*, e.g. mass number or grams/mole for metal units. The cubic lattice constant is in Angstroms. On line 3, Nrho and Nr are the number of tabulated values in the subsequent arrays, drho and dr are the spacing in density and distance space for the values in those arrays, and the specified cutoff becomes the pairwise cutoff used by LAMMPS for the potential. The units of dr are Angstroms; I'm not sure of the units for drho - some measure of electron density.

Following the three header lines are three arrays of tabulated values:

- embedding function F(rho) (Nrho values)
- effective charge function Z(r) (Nr values)
- density function rho(r) (Nr values)

The values for each array can be listed as multiple values per line, so long as each array starts on a new line. For example, the individual Z(r) values are for r = 0,dr,2*dr, ... (Nr-1)*dr.

The units for the embedding function F are eV. The units for the density function rho are the same as for drho (see above, electron density). The units for the effective charge Z are "atomic charge" or sqrt(Hartree * Bohr-radii). For two interacting atoms i,j this is used by LAMMPS to compute the pair potential term in the EAM energy expression as r*phi, in units of eV-Angstroms, via the formula

$$r \cdot \phi = 27.2 \cdot 0.529 \cdot Z_i \cdot Z_j$$

where 1 Hartree = 27.2 eV and 1 Bohr = 0.529 Angstroms.

Style *eam/alloy* computes pairwise interactions using the same formula as style *eam*. However the associated *pair_coeff* command reads a DYNAMO *setfl* file instead of a *funcfl* file. *Setfl* files can be used to model a single-element or alloy system. In the alloy case, as explained above, *setfl* files contain explicit tabulated values for alloy interactions. Thus they allow more generality than *funcfl* files for modeling alloys.

For style *eam/alloy*, potential values are read from a file that is in the DYNAMO multi-element *setfl* format, except that element names (Ni, Cu, etc) are added to one of the lines in the file. If the DYNAMO file was created by a Fortran program, it cannot have "D" values in it for exponents. C only recognizes "e" or "E" for scientific notation.

Only a single pair_coeff command is used with the *eam/alloy* style which specifies a DYNAMO *setfl* file, which contains information for M elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of *setfl* elements to atom types

As an example, the potentials/NiAlH_jea.eam.alloy file is a *setfl* file which has tabulated EAM values for 3 elements and their alloy interactions: Ni, Al, and H. See the *pair_coeff* doc page for alternate ways to specify the path for the potential file. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Ni, and the fourth to be Al, you would use the following pair_coeff command:

```
pair_coeff * * NiAlH_jea.eam.alloy Ni Ni Ni Al
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Ni arguments map LAMMPS atom types 1,2,3 to the Ni element in the *setfl* file. The final Al argument maps LAMMPS atom type 4 to the Al element in the *setfl* file. Note that there is no requirement that your simulation use all the elements specified by the *setfl* file.

If a mapping value is specified as NULL, the mapping is not performed. This can be used when an *eam/alloy* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Setfl files in the *potentials* directory of the LAMMPS distribution have an ".eam.alloy" suffix. A DYNAMO multi-element *setfl* file is formatted as follows:

- lines 1,2,3 = comments (ignored)
- line 4: Nelements Element1 Element2 ... ElementN
- line 5: Nrho, drho, Nr, dr, cutoff

In a DYNAMO *setfl* file, line 4 only lists Nelements = the # of elements in the *setfl* file. For LAMMPS, the element name (Ni, Cu, etc) of each element must be added to the line, in the order the elements appear in the file.

The meaning and units of the values in line 5 is the same as for the *funcfl* file described above. Note that the cutoff (in Angstroms) is a global value, valid for all pairwise interactions for all element pairings.

Following the 5 header lines are Nelements sections, one for each element, each with the following format:

- line 1 = atomic number, mass, lattice constant, lattice type (e.g. FCC)
- embedding function F(rho) (Nrho values)
- density function rho(r) (Nr values)

As with the *funcfl* files, only the mass (in mass *units*, e.g. mass number or grams/mole for metal units) is used by LAMMPS from the first line. The cubic lattice constant is in Angstroms. The F and rho arrays are unique to a single element and have the same format and units as in a *funcfl* file.

Following the Nelements sections, Nr values for each pair potential phi(r) array are listed for all i,j element pairs in the same format as other arrays. Since these interactions are symmetric (i,j = j,i) only phi arrays with i >= j are listed, in the following order: i,j = (1,1), (2,1), (2,2), (3,1), (3,2), (3,3), (4,1), ..., (Nelements, Nelements). Unlike the effective

charge array Z(r) in *funcfl* files, the tabulated values for each phi function are listed in *setfl* files directly as r*phi (in units of eV-Angstroms), since they are for atom pairs.

Style *eam/cd* is similar to the *eam/alloy* style, except that it computes alloy pairwise interactions using the concentration-dependent embedded-atom method (CD-EAM). This model can reproduce the enthalpy of mixing of alloys over the full composition range, as described in *(Stukowski)*. Style *eam/cd/old* is an older, slightly different and slower two-site formulation of the model *(Caro)*.

The pair_coeff command is specified the same as for the *eam/alloy* style. However the DYNAMO *setfl* file must has two lines added to it, at the end of the file:

- line 1: Comment line (ignored)
- line 2: N Coefficient0 Coefficient1 ... CoefficientN

The last line begins with the degree *N* of the polynomial function *h(x)* that modifies the cross interaction between A and B elements. Then *N+1* coefficients for the terms of the polynomial are then listed.

Modified EAM *setfl* files used with the *eam/cd* style must contain exactly two elements, i.e. in the current implementation the *eam/cd* style only supports binary alloys. The first and second elements in the input EAM file are always taken as the *A* and *B* species.

CD-EAM files in the *potentials* directory of the LAMMPS distribution have a ".cdeam" suffix.

Style *eam/fs* computes pairwise interactions for metals and metal alloys using a generalized form of EAM potentials due to Finnis and Sinclair *(Finnis)*. Style *eam/he* is similar to *eam/fs* except that it allows for negative electron density in order to capture the behavior of helium in metals *(Zhou6)*.

The total energy Ei of an atom I is given by

$$E_i = F_\alpha \left(\sum_{j \neq i} \rho_{\alpha\beta}(r_{ij}) \right) + \frac{1}{2} \sum_{j \neq i} \phi_{\alpha\beta}(r_{ij})$$

where $\rho_{\alpha\beta}$ refers to the density contributed by a neighbor atom J of element β at the site of atom I of element α. This has the same form as the EAM formula above, except that rho is now a functional specific to the elements of both atoms I and J, so that different elements can contribute differently to the total electron density at an atomic site depending on the identity of the element at that atomic site.

The associated *pair_coeff* command for style *eam/fs* or *eam/he* reads a DYNAMO *setfl* file that has been extended to include additional $\rho_{\alpha\beta}$ arrays of tabulated values. A discussion of how FS EAM differs from conventional EAM alloy potentials is given in *(Ackland1)*. An example of such a potential is the same author's Fe-P FS potential *(Ackland2)*. Note that while FS potentials always specify the embedding energy with a square root dependence on the total density, the implementation in LAMMPS does not require that; the user can tabulate any functional form desired in the FS potential files.

For style *eam/fs* and *eam/he* the form of the pair_coeff command is exactly the same as for style *eam/alloy*, e.g.

```
pair_coeff * * NiAlH_jea.eam.fs Ni Ni Ni Al
```

with N additional arguments after the filename, where N is the number of LAMMPS atom types. See the *pair_coeff* doc page for alternate ways to specify the path for the potential file. The N values determine the mapping of LAMMPS atom types to EAM elements in the file, as described above for style *eam/alloy*. As with *eam/alloy*, if a mapping value is NULL, the mapping is not performed. This can be used when an *eam/fs* or *eam/he* potential is used as part of a *hybrid* pair style. The NULL values are used as placeholders for atom types that will be used with other potentials.

FS EAM and HE EAM files include more information than the DYNAMO *setfl* format files read by *eam/alloy*, in that i,j density functionals for all pairs of elements are included as needed by the Finnis/Sinclair formulation of the EAM.

FS EAM files in the *potentials* directory of the LAMMPS distribution have an ".eam.fs" suffix. They are formatted as follows:

- lines 1,2,3 = comments (ignored)
- line 4: Nelements Element1 Element2 ... ElementN
- line 5: Nrho, drho, Nr, dr, cutoff

The 5-line header section is identical to an EAM *setfl* file.

Following the header are Nelements sections, one for each element β, each with the following format:

- line 1 = atomic number, mass, lattice constant, lattice type (e.g. FCC)
- embedding function F(rho) (Nrho values)
- density function $\rho_{1\beta}(r)$ for element β at element 1 (Nr values)
- density function $\rho_{2\beta}(r)$ for element β at element 2
- ...
- density function $\rho_{N_{elem}\beta}(r)$ for element β at element N_{elem}

The units of these quantities in line 1 are the same as for *setfl* files. Note that the rho(r) arrays in Finnis/Sinclair can be asymmetric ($\rho_{\alpha\beta}(r) \neq \rho_{\beta\alpha}(r)$) so there are Nelements^2 of them listed in the file.

Following the Nelements sections, Nr values for each pair potential phi(r) array are listed in the same manner (r*phi, units of eV-Angstroms) as in EAM *setfl* files. Note that in Finnis/Sinclair, the phi(r) arrays are still symmetric, so only phi arrays for i >= j are listed.

HE EAM files in the *potentials* directory of the LAMMPS distribution have an ".eam.he" suffix. They are formatted as follows:

- lines 1,2,3 = comments (ignored)
- line 4: Nelements Element1 Element2 ... ElementN
- line 5: Nrho, drho, Nr, dr, cutoff, rhomax

The 5-line header section is identical to an FS EAM file except that line 5 lists an additional value, rhomax. Unlike in FS EAM files where embedding energies F(rho) are always defined between rho = 0 and rho = (Nrho -1)drho, F(rho) in HE EAM files are defined between rho = rhomin and rho = rhomax. Since drho = (rhomax - rhomin)/(Nrho - 1), rhomin = rhomax - (Nrho - 1)drho. The embedding energies F(rho) are listed for rho = rhomin, rhomin + drho, rhomin + 2drho, ..., rhomax. This gives users additional flexibility to define a negative rhomin and therefore an embedding energy function that works for both positive and negative electron densities. The format and units of these sections are identical to the FS EAM files (see above).

Added in version 3Nov2022.

The *eam*, *eam/alloy*, *eam/fs*, and *eam/he* pair styles support extraction of two per-atom quantities by the *fix pair* command. This allows the quantities to be output to files by the *dump* or otherwise processed by other LAMMPS commands.

The names of the two quantities are "rho" and "fp" for the density and derivative of the embedding energy for each atom. Neither quantity needs to be triggered by the *fix pair* command in order for these pair styles to calculate it.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.96.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above with the individual styles. You never need to specify a pair_coeff command with I != J arguments for the eam styles.

This pair style does not support the *pair_modify* shift, table, and tail options.

The eam pair styles do not write their information to *binary restart files*, since it is stored in tabulated potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

The eam pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.96.5 Restrictions

All of these styles are part of the MANYBODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.96.6 Related commands

pair_coeff

4.96.7 Default

none

(Ackland1) Ackland, Condensed Matter (2005).

(Ackland2) Ackland, Mendelev, Srolovitz, Han and Barashev, Journal of Physics: Condensed Matter, 16, S2629 (2004).

(Daw) Daw, Baskes, Phys Rev Lett, 50, 1285 (1983). Daw, Baskes, Phys Rev B, 29, 6443 (1984).

(Finnis) Finnis, Sinclair, Philosophical Magazine A, 50, 45 (1984).

(Zhou6) Zhou, Bartelt, Sills, Physical Review B, 103, 014108 (2021).

(Stukowski) Stukowski, Sadigh, Erhart, Caro; Modeling Simulation Materials Science & Engineering, 7, 075005 (2009).

(Caro) A Caro, DA Crowson, M Caro; Phys Rev Lett, 95, 075702 (2005)

4.97 pair_style edip command

Accelerator Variants: *edip/omp*

4.98 pair_style edip/multi command

4.98.1 Syntax

```
pair_style style
```

- style = *edip* or *edip/multi*

4.98.2 Examples

```
pair_style edip
pair_coeff * * Si.edip Si
```

4.98.3 Description

The *edip* and *edip/multi* styles compute a 3-body *EDIP* potential which is popular for modeling silicon materials where it can have advantages over other models such as the *Stillinger-Weber* or *Tersoff* potentials. The *edip* style has been programmed for single element potentials, while *edip/multi* supports multi-element EDIP runs.

In EDIP, the energy E of a system of atoms is

$$E = \sum_{j \neq i} \phi_2(R_{ij}, Z_i) + \sum_{j \neq i} \sum_{k \neq i, k > j} \phi_3(R_{ij}, R_{ik}, Z_i)$$

$$\phi_2(r, Z) = A \left[\left(\frac{B}{r} \right)^{\rho} - e^{-\beta Z^2} \right] exp\left(\frac{\sigma}{r - a} \right)$$

$$\phi_3(R_{ij}, R_{ik}, Z_i) = exp\left(\frac{\gamma}{R_{ij} - a} \right) exp\left(\frac{\gamma}{R_{ik} - a} \right) h(cos\theta_{ijk}, Z_i)$$

$$Z_i = \sum_{m \neq i} f(R_{im}) \qquad f(r) = \begin{cases} 1 & r < c \\ \exp\left(\frac{\alpha}{1 - x^{-3}} \right) & c < r < a \\ 0 & r > a \end{cases}$$

$$h(l, Z) = \lambda [(1 - e^{-Q(Z)(l + \tau(Z))^2}) + \eta Q(Z)(l + \tau(Z))^2]$$

$$Q(Z) = Q_0 e^{-\mu Z} \qquad \tau(Z) = u_1 + u_2(u_3 e^{-u_4 Z} - e^{-2u_4 Z})$$

where ϕ_2 is a two-body term and ϕ_3 is a three-body term. The summations in the formula are over all neighbors J and K of atom I within a cutoff distance = a. Both terms depend on the local environment of atom I through its effective coordination number defined by Z, which is unity for a cutoff distance < c and gently goes to 0 at distance = a.

Only a single pair_coeff command is used with the *edip* style which specifies a EDIP potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of EDIP elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file Si.edip has EDIP values for Si.

EDIP files in the *potentials* directory of the LAMMPS distribution have a ".edip" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to the two-body and three-body coefficients in the formula above:

- element 1 (the center atom in a 3-body interaction)
- element 2
- element 3
- A (energy units)
- B (distance units)
- cutoffA (distance units)
- cutoffC (distance units)
- α
- β
- η
- γ (distance units)
- *lambda* (energy units)
- μ
- τ
- σ (distance units)
- Q0
- u1
- u2
- u3
- u4

The A, B, beta, sigma parameters are used only for two-body interactions. The eta, gamma, lambda, mu, Q0 and all u1 to u4 parameters are used only for three-body interactions. The alpha and cutoffC parameters are used for the coordination environment function only.

The EDIP potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify EDIP parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

At the moment, only a single element parameterization is implemented. However, the author is not aware of other multi-element EDIP parameterization. If you know any and you are interest in that, please contact the author of the EDIP package.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.98.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.98.5 Restrictions

This pair style can only be used if LAMMPS was built with the MANYBODY package. See the *Build package* doc page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The EDIP potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the EDIP potential with any LAMMPS units, but you would need to create your own EDIP potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.98.6 Related commands

pair_coeff

4.98.7 Default

none

(EDIP) J F Justo et al, Phys Rev B 58, 2539 (1998).

4.99 pair_style eff/cut command

4.99.1 Syntax

```
pair_style eff/cut cutoff keyword args ...
```

- cutoff = global cutoff for Coulombic interactions
- zero or more keyword/value pairs may be appended

  keyword = limit/eradius or pressure/evirials or ecp
    limit/eradius args = none
    pressure/evirials args = none
    ecp args = type element type element ...
      type = LAMMPS atom type (1 to Ntypes)
      element = element symbol (e.g. H, Si)

4.99.2 Examples

```
pair_style eff/cut 39.7
pair_style eff/cut 40.0 limit/eradius
pair_style eff/cut 40.0 limit/eradius pressure/evirials
pair_style eff/cut 40.0 ecp 1 Si 3 C
pair_coeff * *
pair_coeff 2 2 20.0
pair_coeff 1 s 0.320852 2.283269 0.814857
pair_coeff 3 p 22.721015 0.728733 1.103199 17.695345 6.693621
```

4.99.3 Description

This pair style contains a LAMMPS implementation of the electron Force Field (eFF) potential currently under development at Caltech, as described in *(Jaramillo-Botero)*. The eFF for Z<6 was first introduced by *(Su)* in 2007. It has been extended to higher Zs by using effective core potentials (ECPs) that now cover up to second and third row p-block elements of the periodic table.

eFF can be viewed as an approximation to QM wave packet dynamics and Fermionic molecular dynamics, combining the ability of electronic structure methods to describe atomic structure, bonding, and chemistry in materials, and of plasma methods to describe nonequilibrium dynamics of large systems with a large number of highly excited electrons. Yet, eFF relies on a simplification of the electronic wave function in which electrons are described as floating Gaussian wave packets whose position and size respond to the various dynamic forces between interacting classical nuclear particles and spherical Gaussian electron wave packets. The wave function is taken to be a Hartree product of the wave packets. To compensate for the lack of explicit antisymmetry in the resulting wave function, a spin-dependent Pauli potential is included in the Hamiltonian. Substituting this wave function into the time-dependent Schrodinger equation produces equations of motion that correspond - to second order - to classical Hamiltonian relations between electron position and size, and their conjugate momenta. The N-electron wave function is described as a product of one-electron Gaussian functions, whose size is a dynamical variable and whose position is not constrained to a nuclear center. This form allows for straightforward propagation of the wave function, with time, using a simple formulation from which the equations of motion are then integrated with conventional MD algorithms. In addition to this spin-dependent Pauli repulsion potential term between Gaussians, eFF includes the electron kinetic energy from the Gaussians. These two terms are based on first-principles quantum mechanics. On the other hand, nuclei are described as point charges, which interact with other nuclei and electrons through standard electrostatic potential forms.

The full Hamiltonian (shown below), contains then a standard description for electrostatic interactions between a set of delocalized point and Gaussian charges which include, nuclei-nuclei (NN), electron-electron (ee), and nuclei-electron (Ne). Thus, eFF is a mixed QM-classical mechanics method rather than a conventional force field method (in which electron motions are averaged out into ground state nuclear motions, i.e a single electronic state, and particle interactions are described via empirically parameterized interatomic potential functions). This makes eFF uniquely suited to simulate materials over a wide range of temperatures and pressures where electronically excited and ionized states of matter can occur and coexist. Furthermore, the interactions between particles -nuclei and electrons- reduce to the sum of a set of effective pairwise potentials in the eFF formulation. The *eff/cut* style computes the pairwise Coulomb interactions between nuclei and electrons (E_NN,E_Ne,E_ee), and the quantum-derived Pauli (E_PR) and Kinetic energy interactions potentials between electrons (E_KE) for a total energy expression given as,

$$U\left(R,r,s\right)=E_{NN}\left(R\right)+E_{Ne}\left(R,r,s\right)+E_{ee}\left(r,s\right)+E_{KE}\left(r,s\right)+E_{PR}\left(\uparrow\downarrow,S\right)$$

The individual terms are defined as follows:

$$\begin{aligned}
E_{KE} =& \frac{\hbar^2}{m_e}\sum_i \frac{3}{2s_i^2} \\
E_{NN} =& \frac{1}{4\pi\varepsilon_0}\sum_{i<j}\frac{Z_i Z_j}{R_{ij}} \\
E_{Ne} =& -\frac{1}{4\pi\varepsilon_0}\sum_{i,j}\frac{Z_i}{R_{ij}} Erf\left(\frac{\sqrt{2}R_{ij}}{s_j}\right) \\
E_{ee} =& \frac{1}{4\pi\varepsilon_0}\sum_{i<j}\frac{1}{r_{ij}} Erf\left(\frac{\sqrt{2}r_{ij}}{\sqrt{s_i^2+s_j^2}}\right) \\
E_{Pauli} =& \sum_{\sigma_i=\sigma_j} E\left(\uparrow\uparrow\right)_{ij} + \sum_{\sigma_i\neq\sigma_j} E\left(\uparrow\downarrow\right)_{ij}
\end{aligned}$$

where, s_i correspond to the electron sizes, the sigmas i's to the fixed spins of the electrons, Z_i to the charges on the nuclei, R_ij to the distances between the nuclei or the nuclei and electrons, and r_ij to the distances between electrons. For additional details see *(Jaramillo-Botero)*.

The overall electrostatics energy is given in Hartree units of energy by default and can be modified by an energy-conversion constant, according to the units chosen (see *electron_units*). The cutoff Rc, given in Bohrs (by default), truncates the interaction distance. The recommended cutoff for this pair style should follow the minimum image criterion, i.e. half of the minimum unit cell length.

This potential is designed to be used with *atom_style electron* definitions, in order to handle the description of systems with interacting nuclei and explicit electrons.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutoff (distance units)

For *eff/cut*, the cutoff coefficient is optional. If it is not used (as in some of the examples above), the default global value specified in the pair_style command is used.

The *limit/eradius* and *pressure/evirials* keywords are optional. Neither or both must be specified. If not specified they are unset.

The *limit/eradius* keyword is used to restrain electron size from becoming excessively diffuse at very high temperatures were the Gaussian wave packet representation breaks down, and from expanding as free particles to infinite size. If unset, electron radius is free to increase without bounds. If set, a restraining harmonic potential of the form E = 1/2k_ss^2 for s > L_box/2, where k_s = 1 Hartrees/Bohr^2, is applied on the electron radius.

The *pressure/evirials* keyword is used to control between two types of pressure computation: if unset, the computed pressure does not include the electronic radial virials contributions to the total pressure (scalar or tensor). If set, the computed pressure will include the electronic radial virial contributions to the total pressure (scalar and tensor).

The *ecp* keyword is used to associate an ECP representation for a particular atom type. The ECP captures the orbital overlap between a core pseudo particle and valence electrons within the Pauli repulsion. A list of type:element-symbol pairs may be provided for all ECP representations, after the "ecp" keyword.

Note

Default ECP parameters are provided for C, N, O, Al, and Si. Users can modify these using the pair_coeff command as exemplified above. For this, the User must distinguish between two different functional forms supported, one that captures the orbital overlap assuming the s-type core interacts with an s-like valence electron (s-s) and another that assumes the interaction is s-p. For systems that exhibit significant p-character (e.g. C, N, O) the s-p form is recommended. The "s" ECP form requires 3 parameters and the "p" 5 parameters.

Note

There are two different pressures that can be reported for eFF when defining this pair_style, one (default) that considers electrons do not contribute radial virial components (i.e. electrons treated as incompressible 'rigid' spheres) and one that does. The radial electronic contributions to the virials are only tallied if the flexible pressure option is set, and this will affect both global and per-atom quantities. In principle, the true pressure of a system is somewhere in between the rigid and the flexible eFF pressures, but, for most cases, the difference between these two pressures will not be significant over long-term averaged runs (i.e. even though the energy partitioning changes, the total energy remains similar).

Note

This implementation of eFF gives a reasonably accurate description for systems containing nuclei from Z = 1-6 in "all electron" representations. For systems with increasingly non-spherical electrons, Users should use the ECP representations. ECPs are now supported and validated for most of the second and third row elements of the p-block. Predefined parameters are provided for C, N, O, Al, and Si. The ECP captures the orbital overlap between the core and valence electrons (i.e. Pauli repulsion) with one of the functional forms:

$$E_{Pauli(ECP_s)} = p_1 \exp\left(-\frac{p_2 r^2}{p_3 + s^2}\right)$$

$$E_{Pauli(ECP_p)} = p_1 \left(\frac{2}{p_2/s + s/p_2}\right)(r - p_3 s)^2 \exp\left[-\frac{p_4 (r - p_3 s)^2}{p_5 + s^2}\right]$$

Where the first form correspond to core interactions with s-type valence electrons and the second to core interactions with p-type valence electrons.

The current version adds full support for models with fixed-core and ECP definitions. to enable larger timesteps (i.e. by avoiding the high frequency vibrational modes -translational and radial- of the 2 s electrons), and in the ECP case to reduce the increased orbital complexity in higher Z elements (up to Z<18). A fixed-core should be defined with a mass that includes the corresponding nuclear mass plus the 2 s electrons in atomic mass units (2x5.4857990943e-4), and a radius equivalent to that of minimized 1s electrons (see examples under /examples/PACKAGES/eff/fixed-core). An pseudo-core should be described with a mass that includes the corresponding nuclear mass, plus all the core electrons

(i.e no outer shell electrons), and a radius equivalent to that of a corresponding minimized full-electron system. The charge for a pseudo-core atom should be given by the number of outer shell electrons.

In general, eFF excels at computing the properties of materials in extreme conditions and tracing the system dynamics over multi-picosecond timescales; this is particularly relevant where electron excitations can change significantly the nature of bonding in the system. It can capture with surprising accuracy the behavior of such systems because it describes consistently and in an unbiased manner many different kinds of bonds, including covalent, ionic, multicenter, ionic, and plasma, and how they interconvert and/or change when they become excited. eFF also excels in computing the relative thermochemistry of isodemic reactions and conformational changes, where the bonds of the reactants are of the same type as the bonds of the products. eFF assumes that kinetic energy differences dominate the overall exchange energy, which is true when the electrons present are nearly spherical and nodeless and valid for covalent compounds such as dense hydrogen, hydrocarbons, and diamond; alkali metals (e.g. lithium), alkali earth metals (e.g. beryllium) and semimetals such as boron; and various compounds containing ionic and/or multicenter bonds, such as boron dihydride.

4.99.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the cutoff distance for the *eff/cut* style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

The *pair_modify* shift option is not relevant for these pair styles.

The *eff/long* (not yet available) style supports the *pair_modify* table option for tabulation of the short-range portion of the long-range Coulombic interaction.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.99.5 Restrictions

These pair styles will only be enabled if LAMMPS is built with the EFF package. It will only be enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles require that particles store electron attributes such as radius, radial velocity, and radial force, as defined by the *atom_style*. The *electron* atom style does all of this.

Thes pair styles require you to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

4.99.6 Related commands

pair_coeff

4.99.7 Default

If not specified, limit_eradius = 0 and pressure_with_evirials = 0.

(Su) Su and Goddard, Excited Electron Dynamics Modeling of Warm Dense Matter, Phys Rev Lett, 99:185003 (2007).

(Jaramillo-Botero) Jaramillo-Botero, Su, Qi, Goddard, Large-scale, Long-term Non-adiabatic Electron Molecular Dynamics for Describing Material Properties and Phenomena in Extreme Environments, J Comp Chem, 32, 497-512 (2011).

4.100 pair_style eim command

Accelerator Variants: *eim/omp*

4.100.1 Syntax

```
pair_style style
```

- style = *eim*

4.100.2 Examples

```
pair_style eim
pair_coeff * * Na Cl ../potentials/ffield.eim Na Cl
pair_coeff * * Na Cl ffield.eim  Na Na Na Cl
pair_coeff * * Na Cl ../potentials/ffield.eim Cl NULL Na
```

4.100.3 Description

Style *eim* computes pairwise interactions for ionic compounds using embedded-ion method (EIM) potentials *(Zhou)*. The energy of the system E is given by

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=i_1}^{i_N}\phi_{ij}\left(r_{ij}\right) + \sum_{i=1}^{N}E_i\left(q_i,\sigma_i\right)$$

The first term is a double pairwise sum over the J neighbors of all I atoms, where ϕ_{ij} is a pair potential. The second term sums over the embedding energy E_i of atom I, which is a function of its charge q_i and the electrical potential σ_i at its location. E_i, q_i, and $sigma_i$ are calculated as

$$q_i = \sum_{j=i_1}^{i_N}\eta_{ji}\left(r_{ij}\right)$$

$$\sigma_i = \sum_{j=i_1}^{i_N} q_j \cdot \psi_{ij}\left(r_{ij}\right)$$

$$E_i\left(q_i,\sigma_i\right) = \frac{1}{2}\cdot q_i \cdot \sigma_i$$

where η_{ji} is a pairwise function describing electron flow from atom I to atom J, and ψ_{ij} is another pairwise function. The multi-body nature of the EIM potential is a result of the embedding energy term. A complete list of all the pair functions used in EIM is summarized below

$$\phi_{ij}(r) = \begin{cases} \left[\frac{E_{b,ij}\beta_{ij}}{\beta_{ij}-\alpha_{ij}} \exp\left(-\alpha_{ij}\frac{r-r_{e,ij}}{r_{e,ij}}\right) - \frac{E_{b,ij}\alpha_{ij}}{\beta_{ij}-\alpha_{ij}} \exp\left(-\beta_{ij}\frac{r-r_{e,ij}}{r_{e,ij}}\right)\right] f_c\left(r, r_{e,ij}, r_{c,\phi,ij}\right), & p_{ij} = 1 \\ \left[\frac{E_{b,ij}\beta_{ij}}{\beta_{ij}-\alpha_{ij}} \left(\frac{r_{e,ij}}{r}\right)^{\alpha_{ij}} - \frac{E_{b,ij}\alpha_{ij}}{\beta_{ij}-\alpha_{ij}} \left(\frac{r_{e,ij}}{r}\right)^{\beta_{ij}}\right] f_c\left(r, r_{e,ij}, r_{c,\phi,ij}\right), & p_{ij} = 2 \end{cases}$$

$$\eta_{ji} = A_{\eta,ij}\left(\chi_j - \chi_i\right) f_c\left(r, r_{s,\eta,ij}, r_{c,\eta,ij}\right)$$

$$\psi_{ij}(r) = A_{\psi,ij} \exp\left(-\zeta_{ij} r\right) f_c\left(r, r_{s,\psi,ij}, r_{c,\psi,ij}\right)$$

$$f_c\left(r, r_p, r_c\right) = 0.510204 \cdot \operatorname{erfc}\left[\frac{1.64498\left(2r - r_p - r_c\right)}{r_c - r_p}\right] - 0.010204$$

Here $E_b, r_e, r_(c,\phi), \alpha, \beta, A_(\psi), \zeta, r_(s,\psi), r_(c,\psi), A_(\eta), r_(s,\eta), r_(c,\eta), \chi$, and pair function type p are parameters, with subscripts ij indicating the two species of atoms in the atomic pair.

Note

Even though the EIM potential is treating atoms as charged ions, you should not use a LAMMPS *atom_style* that stores a charge on each atom and thus requires you to assign a charge to each atom, e.g. the *charge* or *full* atom styles. This is because the EIM potential infers the charge on an atom from the equation above for q_i; you do not assign charges explicitly.

All the EIM parameters are listed in a potential file which is specified by the *pair_coeff* command. This is an ASCII text file in a format described below. The "ffield.eim" file included in the "potentials" directory of the LAMMPS distribution currently includes nine elements Li, Na, K, Rb, Cs, F, Cl, Br, and I. A system with any combination of these elements can be modeled. This file is parameterized in terms of LAMMPS *metal units*.

Note that unlike other potentials, cutoffs for EIM potentials are not set in the pair_style or pair_coeff command; they are specified in the EIM potential file itself. Likewise, the EIM potential file lists atomic masses; thus you do not need to use the *mass* command to specify them.

Only a single pair_coeff command is used with the *eim* style which specifies an EIM potential file and the element(s) to extract information for. The EIM elements are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- Elem1, Elem2, ...
- EIM potential file
- N element names = mapping of EIM elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example like one of those above, suppose you want to model a system with Na and Cl atoms. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Na, and the fourth to be Cl, you would use the following pair_coeff command:

```
pair_coeff * * Na Cl ffield.eim Na Na Na Cl
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The filename is the EIM potential file. The Na and Cl arguments (before the file name) are the two elements for which info will be extracted from the potential file. The first three trailing Na arguments map LAMMPS atom types 1,2,3 to the EIM Na element. The final Cl argument maps LAMMPS atom type 4 to the EIM Cl element.

If a mapping value is specified as NULL, the mapping is not performed. This can be used when an *eim* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The ffield.eim file in the *potentials* directory of the LAMMPS distribution is formatted as follows:

Lines starting with # are comments and are ignored by LAMMPS. Lines starting with "global:" include three global values. The first value divides the cations from anions, i.e., any elements with electronegativity above this value are viewed as anions, and any elements with electronegativity below this value are viewed as cations. The second and third values are related to the cutoff function - i.e. the 0.510204, 1.64498, and 0.010204 shown in the above equation can be derived from these values.

Lines starting with "element:" are formatted as follows: name of element, atomic number, atomic mass, electronic negativity, atomic radius (LAMMPS ignores it), ionic radius (LAMMPS ignores it), cohesive energy (LAMMPS ignores it), and q0 (must be 0).

Lines starting with "pair:" are entered as: element 1, element 2, r_(c,phi), r_(c,phi) (redundant for historical reasons), E_b, r_e, alpha, beta, r_(c,eta), A_(eta), r_(s,eta), r_(c,psi), A_(psi), zeta, r_(s,psi), and p.

The lines in the file can be in any order; LAMMPS extracts the info it needs.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.100.4 Restrictions

This style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package.

4.100.5 Related commands

pair_coeff

4.100.6 Default

none

(Zhou) Zhou, submitted for publication (2010). Please contact Xiaowang Zhou (Sandia) for details via email at xzhou at sandia.gov.

4.101 pair_style exp6/rx command

Accelerator Variants: *exp6/rx/kk*

4.101.1 Syntax

```
pair_style exp6/rx cutoff ...
```

- cutoff = global cutoff for DPD interactions (distance units)
- weighting = fractional or molecular (optional)

4.101.2 Examples

```
pair_style exp6/rx 10.0
pair_style exp6/rx 10.0 fractional
pair_style exp6/rx 10.0 molecular
pair_coeff * * exp6.params h2o h2o exponent 1.0 1.0 10.0
pair_coeff * * exp6.params h2o 1fluid exponent 1.0 1.0 10.0
pair_coeff * * exp6.params 1fluid 1fluid exponent 1.0 1.0 10.0
pair_coeff * * exp6.params 1fluid 1fluid none 10.0
pair_coeff * * exp6.params 1fluid 1fluid polynomial filename 10.0
```

4.101.3 Description

Style *exp6/rx* is used in reaction DPD simulations, where the coarse-grained (CG) particles are composed of m species whose reaction rate kinetics are determined from a set of n reaction rate equations through the *fix rx* command. The species of one CG particle can interact with a species in a neighboring CG particle through a site-site interaction potential model. The *exp6/rx* style computes an exponential-6 potential given by

$$U_{ij}(r) = \frac{\varepsilon}{\alpha - 6}\{6\exp[\alpha(1 - \frac{r_{ij}}{R_m})] - \alpha(\frac{R_m}{r_{ij}})^6\}$$

where the ε parameter determines the depth of the potential minimum located at R_m, and α determines the softness of the repulsion.

The coefficients must be defined for each species in a given particle type via the *pair_coeff* command as in the examples above, where the first argument is the filename that includes the exponential-6 parameters for each species. The file includes the species tag followed by the α, ε and R_m parameters. The format of the file is described below.

The second and third arguments specify the site-site interaction potential between two species contained within two different particles. The species tags must either correspond to the species defined in the reaction kinetics files specified with the *fix rx* command or they must correspond to the tag "1fluid", signifying interaction with a product species mixture determined through a one-fluid approximation. The interaction potential is weighted by the geometric average of either the mole fraction concentrations or the number of molecules associated with the interacting coarse-grained particles (see the *fractional* or *molecular* weighting pair style options). The coarse-grained potential is stored before and after the reaction kinetics solver is applied, where the difference is defined to be the internal chemical energy (uChem).

The fourth argument specifies the type of scaling that will be used to scale the EXP-6 parameters as reactions occur. Currently, there are three scaling options: *exponent*, *polynomial* and *none*.

Exponent scaling requires two additional arguments for scaling the R_m and ε parameters, respectively. The scaling factor is computed by phi^exponent, where phi is the number of molecules represented by the coarse-grain particle and exponent is specified as a pair coefficient argument for R_m and ε, respectively. The R_m and ε parameters are multiplied by the scaling factor to give the scaled interaction parameters for the CG particle.

Polynomial scaling requires a filename to be specified as a pair coeff argument. The file contains the coefficients to a fifth order polynomial for the α, ε and R_m parameters that depend upon phi (the number of molecules represented by the CG particle). The format of a polynomial file is provided below.

The *none* option to the scaling does not have any additional pair coeff arguments. This is equivalent to specifying the *exponent* option with R_m and ε exponents of 0.0 and 0.0, respectively.

The final argument specifies the interaction cutoff (optional).

The format of a tabulated file is as follows (without the parenthesized comments):

```
# exponential-6 parameters for various species      (one or more comment or blank lines)

h2o  exp6  11.00 0.02 3.50                          (species, exp6, alpha, Rm, epsilon)
no2  exp6  13.60 0.01 3.70
...
co2  exp6  13.00 0.03 3.20
```

The format of the polynomial scaling file as follows (without the parenthesized comments):

```
# POLYNOMIAL FILE          (one or more comment or blank lines)

#  General Functional Form:
#  A*phi^5 + B*phi^4 + C*phi^3 + D*phi^2 + E*phi + F
#
#  Parameter  A        B          C         D         E         F
                      (blank)
alpha        0.0000   0.00000    0.00008   0.04955   -0.73804  13.63201
epsilon      0.0000   0.00478    -0.06283  0.24486   -0.33737  2.60097
rm           0.0001   -0.00118   -0.00253  0.05812   -0.00509  1.50106
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections.

Following a blank line, the next N lines list the species and their corresponding parameters. The first argument is the species tag, the second argument is the exp6 tag, the third argument is the α parameter (energy units), the fourth argument is the ε parameter (energy-distance^6 units), and the fifth argument is the R_m parameter (distance units). If a species tag of "1fluid" is listed as a pair coefficient, a one-fluid approximation is specified where a concentration-dependent combination of the parameters is computed through the following equations:

$$R_m^3 = \sum_a \sum_b x_a x_b R_{m,ab}^3$$

$$\varepsilon = \frac{1}{R_m^3} \sum_a \sum_b x_a x_b \varepsilon_{ab} R_{m,ab}^3$$

$$\alpha = \frac{1}{\varepsilon R_m^3} \sum_a \sum_b x_a x_b \alpha_{ab} \varepsilon_{ab} R_{m,ab}^3$$

where

$$\varepsilon_{ab} = \sqrt{\varepsilon_a \varepsilon_b}$$

$$R_{m,ab} = \frac{R_{m,a} + R_{m,b}}{2}$$

$$\alpha_{ab} = \sqrt{\alpha_a \alpha_b}$$

and x_a and x_b are the mole fractions of a and b, respectively, which comprise the gas mixture.

4.101.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift option for the energy of the exp() and 1/r^6 portion of the pair interaction.

This style does not support the pair_modify tail option for adding long-range tail corrections to energy and pressure for the A,C terms in the pair interaction.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.101.5 Restrictions

This command is part of the DPD-REACT package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.101.6 Related commands

pair_coeff

4.101.7 Default

fractional weighting

4.102 pair_style extep command

4.102.1 Syntax

```
pair_style extep
```

4.102.2 Examples

```
pair_style extep
pair_coeff * * BN.extep B N
```

4.102.3 Description

Style *extep* computes the Extended Tersoff Potential (ExTeP) interactions as described in *(Los2017)*.

4.102.4 Restrictions

none

4.102.5 Related commands

pair_tersoff

4.102.6 Default

none

(Los2017) J. H. Los et al. "Extended Tersoff potential for boron nitride: Energetics and elastic properties of pristine and defective h-BN", Phys. Rev. B 96 (184108), 2017.

4.103 pair_style lj/cut/soft command

Accelerator Variants: *lj/cut/soft/omp*

4.104 pair_style lj/cut/coul/cut/soft command

Accelerator Variants: *lj/cut/coul/cut/soft/gpu*, *lj/cut/coul/cut/soft/omp*

4.105 pair_style lj/cut/coul/long/soft command

Accelerator Variants: *lj/cut/coul/long/soft/gpu*, *lj/cut/coul/long/soft/omp*

4.106 pair_style lj/cut/tip4p/long/soft command

Accelerator Variants: *lj/cut/tip4p/long/soft/omp*

4.107 pair_style lj/charmm/coul/long/soft command

Accelerator Variants: *lj/charmm/coul/long/soft/omp*

4.108 pair_style lj/class2/soft command

4.109 pair_style lj/class2/coul/cut/soft command

4.110 pair_style lj/class2/coul/long/soft command

4.111 pair_style coul/cut/soft command

Accelerator Variants: *coul/cut/soft/omp*

4.112 pair_style coul/long/soft command

Accelerator Variants: *coul/long/soft/omp*

4.113 pair_style tip4p/long/soft command

Accelerator Variants: *tip4p/long/soft/omp*

4.114 pair_style morse/soft command

4.114.1 Syntax

```
pair_style style args
```

- style = *lj/cut/soft* or *lj/cut/coul/cut/soft* or *lj/cut/coul/long/soft* or *lj/cut/tip4p/long/soft* or *lj/charmm/coul/long/soft* or *lj/class2/soft* or *lj/class2/coul/cut/soft* or *lj/class2/coul/long/soft* or *coul/cut/soft* or *coul/long/soft* or *tip4p/long/soft* or *morse/soft*
- args = list of arguments for a particular style

lj/cut/soft args = n alpha_lj cutoff
  n, alpha_LJ = parameters of soft-core potential
  cutoff = global cutoff for Lennard-Jones interactions (distance units)
lj/cut/coul/cut/soft args = n alpha_LJ alpha_C cutoff (cutoff2)
  n, alpha_LJ, alpha_C = parameters of soft-core potential

cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/coul/long/soft args = n alpha_LJ alpha_C cutoff
n, alpha_LJ, alpha_C = parameters of the soft-core potential
cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/tip4p/long/soft args = otype htype btype atype qdist n alpha_LJ alpha_C cutoff (cutoff2)
otype,htype = atom types (numeric or type label) for TIP4P O and H
btype,atype = bond and angle types (numeric or type label) for TIP4P waters
qdist = distance from O atom to massless charge (distance units)
n, alpha_LJ, alpha_C = parameters of the soft-core potential
cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/charmm/coul/long/soft args = n alpha_LJ alpha_C inner outer (cutoff)
n, alpha_LJ, alpha_C = parameters of the soft-core potential
inner, outer = global switching cutoffs for LJ (and Coulombic if only 5 args)
cutoff = global cutoff for Coulombic (optional, outer is Coulombic cutoff if only 5 args)
lj/class2/soft args = n alpha_lj cutoff
n, alpha_LJ = parameters of soft-core potential
cutoff = global cutoff for Lennard-Jones interactions (distance units)
lj/class2/coul/cut/soft args = n alpha_LJ alpha_C cutoff (cutoff2)
n, alpha_LJ, alpha_C = parameters of soft-core potential
cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/class2/coul/long/soft args = n alpha_LJ alpha_C cutoff (cutoff2)
n, alpha_LJ, alpha_C = parameters of soft-core potential
cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
cutoff2 = global cutoff for Coulombic (optional) (distance units)
coul/cut/soft args = n alpha_C cutoff
n, alpha_C = parameters of the soft-core potential
cutoff = global cutoff for Coulomb interactions (distance units)
coul/long/soft args = n alpha_C cutoff
n, alpha_C = parameters of the soft-core potential
cutoff = global cutoff for Coulomb interactions (distance units)
tip4p/long/soft args = otype htype btype atype qdist n alpha_C cutoff
otype,htype = atom types (numeric or type label) for TIP4P O and H
btype,atype = bond and angle types (numeric or type label) for TIP4P waters
qdist = distance from O atom to massless charge (distance units)
n, alpha_C = parameters of the soft-core potential
cutoff = global cutoff for Coulomb interactions (distance units)
morse/soft args = n lf cutoff
n = soft-core parameter
lf = transformation range is lf $<$ lambda $<$ 1
cutoff = global cutoff for Morse interactions (distance units)

4.114.2 Examples

```
pair_style lj/cut/soft 2.0 0.5 9.5
pair_coeff * * 0.28 3.1 1.0
pair_coeff 1 1 0.28 3.1 1.0 9.5

pair_style lj/cut/coul/cut/soft 2.0 0.5 10.0 9.5
pair_style lj/cut/coul/cut/soft 2.0 0.5 10.0 9.5 9.5
pair_coeff * * 0.28 3.1 1.0
pair_coeff 1 1 0.28 3.1 0.5 10.0
pair_coeff 1 1 0.28 3.1 0.5 10.0 9.5

pair_style lj/cut/coul/long/soft 2.0 0.5 10.0 9.5
pair_style lj/cut/coul/long/soft 2.0 0.5 10.0 9.5 9.5
pair_coeff * * 0.28 3.1 1.0
pair_coeff 1 1 0.28 3.1 0.0 10.0
pair_coeff 1 1 0.28 3.1 0.0 10.0 9.5

pair_style lj/cut/tip4p/long/soft 1 2 7 8 0.15 2.0 0.5 10.0 9.8
pair_style lj/cut/tip4p/long/soft 1 2 7 8 0.15 2.0 0.5 10.0 9.8 9.5
pair_coeff * * 0.155 3.1536 1.0
pair_coeff 1 1 0.155 3.1536 1.0 9.5

pair_style lj/cut/tip4p/long/soft OW HW HW-OW HW-OW-HW 0.15 2.0 0.5 10.0 9.8
labelmap atom 1 OW 2 HW
labelmap bond 1 HW-OW
labelmap angle 1 HW-OW-HW
pair_coeff * * 0.155 3.1536 1.0
pair_coeff OW OW 0.155 3.1536 1.0 9.5

pair_style lj/charmm/coul/long 2.0 0.5 10.0 8.0 10.0
pair_style lj/charmm/coul/long 2.0 0.5 10.0 8.0 10.0 9.0
pair_coeff * * 0.28 3.1 1.0
pair_coeff 1 1 0.28 3.1 1.0 0.14 3.1

pair_style lj/class2/coul/long/soft 2.0 0.5 10.0 9.5
pair_style lj/class2/coul/long/soft 2.0 0.5 10.0 9.5 9.5
pair_coeff * * 0.28 3.1 1.0
pair_coeff 1 1 0.28 3.1 0.0 10.0
pair_coeff 1 1 0.28 3.1 0.0 10.0 9.5

pair_style coul/long/soft 1.0 10.0 9.5
pair_coeff * * 1.0
pair_coeff 1 1 1.0

pair_style tip4p/long/soft 1 2 7 8 0.15 2.0 0.5 10.0 9.8
pair_coeff * * 1.0
pair_coeff 1 1 1.0

pair_style morse/soft 4 0.9 10.0
pair_coeff * * 100.0 2.0 1.5 1.0
pair_coeff 1 1 100.0 2.0 1.5 1.0 3.0
```

Example input scripts available: examples/PACKAGES/fep

4.114.3 Description

These pair styles have a soft repulsive core, tunable by a parameter lambda, in order to avoid singularities during free energy calculations when sites are created or annihilated *(Beutler)*. When lambda tends to 0 the pair interaction vanishes with a soft repulsive core. When lambda tends to 1, the pair interaction approaches the normal, non-soft potential. These pair styles are suited for "alchemical" free energy calculations using the *fix adapt/fep* and *compute fep* commands.

The *lj/cut/soft* style and related sub-styles compute the 12-6 Lennard-Jones and Coulomb potentials modified by a soft core, with the functional form

$$E = \lambda^n 4\varepsilon \left\{ \frac{1}{\left[\alpha_{\mathrm{LJ}}(1-\lambda)^2 + \left(\frac{r}{\sigma}\right)^6\right]^2} - \frac{1}{\alpha_{\mathrm{LJ}}(1-\lambda)^2 + \left(\frac{r}{\sigma}\right)^6} \right\} \qquad r < r_c$$

The *lj/class2/soft* style is a 9-6 potential with the exponent of the denominator of the first term in brackets taking the value 1.5 instead of 2 (other details differ, see the form of the potential in *pair_style lj/class2*).

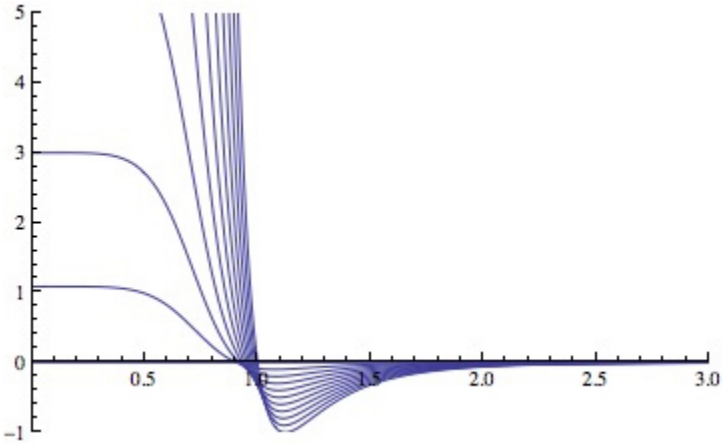
Coulomb interactions can also be damped with a soft core at short distance,

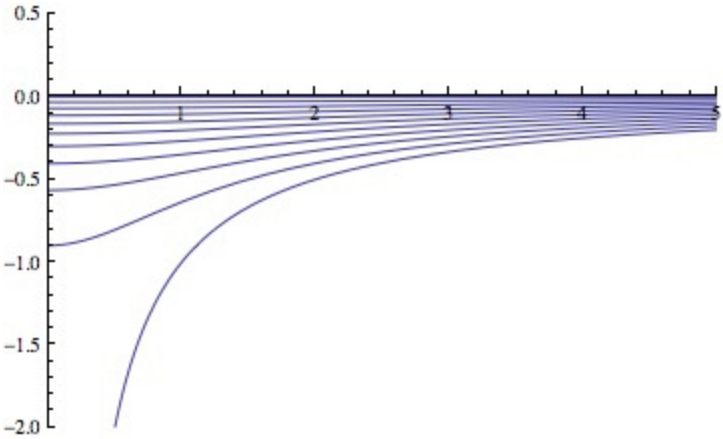
$$E = \lambda^n \frac{C q_i q_j}{\varepsilon \left[\alpha_{\mathrm{C}}(1-\lambda)^2 + r^2\right]^{1/2}} \qquad r < r_c$$

In the Coulomb part C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, and epsilon is the dielectric constant which can be set by the *dielectric* command.

The coefficient lambda is an activation parameter. When $\lambda = 1$ the pair potential is identical to a Lennard-Jones term or a Coulomb term or a combination of both. When $\lambda = 0$ the interactions are deactivated. The transition between these two extrema is smoothed by a soft repulsive core in order to avoid singularities in potential energy and forces when sites are created or annihilated and can overlap *(Beutler)*.

The parameters n, α_{LJ} and α_{C} are set in the *pair_style* command, before the cutoffs. Usual choices for the exponent are $n = 2$ or $n = 1$. For the remaining coefficients $\alpha_{\mathrm{LJ}} = 0.5$ and $\alpha_{\mathrm{C}} = 10$ A^2 are appropriate choices. Plots of the 12-6 LJ and Coulomb terms are shown below, for lambda ranging from 1 to 0 every 0.1.





For the *lj/cut/coul/cut/soft* or *lj/cut/coul/long/soft* pair styles, as well as for the equivalent *class2* versions, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ϵ (energy units)
- σ (distance units)
- λ (activation parameter, between 0 and 1)
- cutoff1 (distance units)
- cutoff2 (distance units)

The latter two coefficients are optional. If not specified, the global LJ and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both LJ and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the LJ and Coulombic cutoffs for this type pair. You cannot specify 2 cutoffs for style *lj/cut/soft*, since it has no Coulombic terms. For the *coul/cut/soft* and *coul/long/soft* only lambda and the optional cutoff2 are to be specified.

Style *lj/cut/tip4p/long/soft* implements a soft-core version of the TIP4P water model. The usage of the TIP4P pair style is documented in the *pair_lj* styles. In the soft version the parameters n, α_{LJ} and α_C are set in the *pair_style* command, after the specific parameters of the TIP4P water model and before the cutoffs. The activation parameter lambda is supplied as an argument of the *pair_coeff* command, after epsilon and sigma and before the optional cutoffs.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

Style *lj/charmm/coul/long/soft* implements a soft-core version of the modified 12-6 LJ potential used in CHARMM and documented in the *pair_style lj/charmm/coul/long* style. In the soft version the parameters n, α_{LJ} and α_C are set in the *pair_style* command, before the global cutoffs. The activation parameter lambda is introduced as an argument of the *pair_coeff* command, after ϵ and σ and before the optional eps14 and sigma14.

Style *lj/class2/soft* implements a soft-core version of the 9-6 potential in *pair_style lj/class2*. In the soft version the parameters n, α_{LJ} and α_C are set in the *pair_style* command, before the global cutoffs. The activation parameter lambda is introduced as an argument of the the *pair_coeff* command, after ϵ and σ and before the optional cutoffs.

The *coul/cut/soft*, *coul/long/soft* and *tip4p/long/soft* sub-styles are designed to be combined with other pair potentials via the *pair_style hybrid/overlay* command. This is because they have no repulsive core. Hence, if used by themselves, there will be no repulsion to keep two oppositely charged particles from overlapping each other. In this case, if $\lambda = 1$, a singularity may occur. These sub-styles are suitable to represent charges embedded in the Lennard-Jones radius

of another site (for example hydrogen atoms in several water models). The λ must be defined for each pair, and *coul/cut/soft* can accept an optional cutoff as the second coefficient.

Note

When using the soft-core Coulomb potentials with long-range solvers (*coul/long/soft*, *lj/cut/coul/long/soft*, etc.) in a free energy calculation in which sites holding electrostatic charges are being created or annihilated (using *fix adapt/fep* and *compute fep*) it is important to adapt both the λ activation parameter (from 0 to 1, or the reverse) and the value of the charge (from 0 to its final value, or the reverse). This ensures that long-range electrostatic terms (kspace) are correct. It is not necessary to use soft-core Coulomb potentials if the van der Waals site is present during the free-energy route, thus avoiding overlap of the charges. Examples are provided in the LAMMPS source directory tree, under examples/PACKAGES/fep.

Note

To avoid division by zero do not set $\sigma = 0$ in the *lj/cut/soft* and related styles; use the lambda parameter instead to activate/deactivate interactions, or use $\varepsilon = 0$ and $\sigma = 1$. Alternatively, when sites do not interact though the Lennard-Jones term the *coul/long/soft* or similar sub-style can be used via the *pair_style hybrid/overlay* command.

The *morse/soft* variant modifies the *pair_morse* style at short range to have a soft core. The functional form differs from that of the *lj/soft* styles, and is instead given by:

$$
\begin{aligned}
s(\lambda) =& (1-\lambda)/(1-\lambda_f), \qquad B = -2De^{-2\alpha r_0}(e^{\alpha r_0}-1)/3 \\
E =& D_0\left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}\right] + s(\lambda)Be^{-3\alpha(r-r_0)}, && \lambda \geq \lambda_f, \quad r < r_c \\
E =& \left(D_0\left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}\right] + Be^{-3\alpha(r-r_0)}\right)(\lambda/\lambda_f)^n, && \lambda < \lambda_f, \quad r < r_c
\end{aligned}
$$

The *morse/soft* style requires the following pair coefficients:

- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)
- λ (unitless, between 0.0 and 1.0)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global morse cutoff is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.114.4 Mixing, shift, table, tail correction, restart, rRESPA info

The different versions of the *lj/cut/soft* pair styles support mixing. For atom type pairs I,J and I != J, the ϵ and σ coefficients and cutoff distance for these pair styles can be mixed. The default mix value is *geometric* for 12-6 styles.

The mixing rule for epsilon and sigma for *lj/class2/soft* 9-6 potentials is to use the *sixthpower* formulas. The *pair_modify mix* setting is thus ignored for class2 potentials for ϵ and σ. However it is still followed for mixing the cutoff distance. See the *pair_modify* command for details.

The *morse/soft* pair style does not support mixing. Thus, coefficients for all LJ pairs must be specified explicitly.

All of the pair styles with soft core support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

The different versions of the *lj/cut/soft* pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure for the Lennard-Jones portion of the pair interaction.

Note

The analytical form of the tail corrections for energy and pressure used in the *lj/cut/soft* potentials are approximate, being identical to that of the corresponding non-soft potentials scaled by a factor λ^n. The errors due to this approximation should be negligible. For example, for a cutoff of 2.5σ this approximation leads to maximum relative errors in tail corrections of the order of 1e-4 for energy and virial ($\alpha_{\mathrm{LJ}} = 0.5, n = 2$). The error vanishes when lambda approaches 0 or 1. Note that these are the errors affecting the long-range tail (itself a correction to the interaction energy) which includes other approximations, namely that the system is homogeneous (local density equal the average density) beyond the cutoff.

The *morse/soft* pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

All of these pair styles write information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

4.114.5 Restrictions

The pair styles with soft core are only enabled if LAMMPS was built with the FEP package. The *long* versions also require the KSPACE package to be installed. The soft *tip4p* versions also require the MOLECULE package to be installed. These styles are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

4.114.6 Related commands

pair_coeff, *fix adapt*, *fix adapt/fep*, *compute fep*

4.114.7 Default

none

(Beutler) Beutler, Mark, van Schaik, Gerber, van Gunsteren, Chem Phys Lett, 222, 529 (1994).

4.115 pair_style gauss command

Accelerator Variants: *gauss/gpu*, *gauss/omp*

4.116 pair_style gauss/cut command

Accelerator Variants: *gauss/cut/omp*

4.116.1 Syntax

```
pair_style gauss cutoff
pair_style gauss/cut cutoff
```

- cutoff = global cutoff for Gauss interactions (distance units)

4.116.2 Examples

```
pair_style gauss 12.0
pair_coeff * * 1.0 0.9
pair_coeff 1 4 1.0 0.9 10.0

pair_style gauss/cut 3.5
pair_coeff 1 4 0.2805 1.45 0.112
```

4.116.3 Description

Style *gauss* computes a tethering potential of the form

$$E = -A \exp(-Br^2) \qquad r < r_c$$

between an atom and its corresponding tether site which will typically be a frozen atom in the simulation. r_c is the cutoff.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (energy units)
- B (1/distance^2 units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

Style *gauss/cut* computes a generalized Gaussian interaction potential between pairs of particles:

$$E = \frac{H}{\sigma_h \sqrt{2\pi}} \exp\left[-\frac{(r - r_{mh})^2}{2\sigma_h^2}\right]$$

where H determines together with the standard deviation σ_h the peak height of the Gaussian function, and r_{mh} the peak position. Examples of the use of the Gaussian potentials include implicit solvent simulations of salt ions *(Lenart)* and of surfactants *(Jusufi)*. In these instances the Gaussian potential mimics the hydration barrier between a pair of particles. The hydration barrier is located at r_{mh} and has a width of σ_h. The prefactor determines the height of the potential barrier.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- H (energy * distance units)
- r_{mh} (distance units)
- σ_h (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.116.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A, B, H, sigma_h, r_mh parameters, and the cutoff distance for these pair styles can be mixed:

- A (energy units)
- $\sqrt{\frac{1}{B}}$ (distance units, see below)
- H (energy units)
- r_{mh} (distance units)
- σ_h (distance units)
- cutoff (distance units)

The default mix value is *geometric*. Only *arithmetic* and *geometric* mix values are supported. See the "pair_modify" command for details.

The A and H parameters are mixed using the same rules normally used to mix the "epsilon" parameter in a Lennard Jones interaction. The sigma_h, r_mh, and the cutoff distance are mixed using the same rules used to mix the "sigma" parameter in a Lennard Jones interaction. The B parameter is converted to a distance (sigma), before mixing (using sigma=B^-0.5), and converted back to a coefficient afterwards (using B=sigma^2). Negative A values are converted to positive A values (using abs(A)) before mixing, and converted back after mixing (by multiplying by min(sign(Ai),sign(Aj))). This way, if either particle is repulsive (if Ai<0 or Aj<0), then the default interaction between both particles will be repulsive.

For the *gauss* style there is no effect due to the Gaussian well beyond the cutoff; hence reasonable cutoffs need to be specified.

The *gauss/cut* style supports the *pair_modify* shift option for the energy of the Gauss-potential portion of the pair interaction.

The *pair_modify* table and tail options are not relevant for these pair styles.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

The *gauss* pair style tallies an "occupancy" count of how many Gaussian-well sites have an atom within the distance at which the force is a maximum = sqrt(0.5/b). This quantity can be accessed via the *compute pair* command as a vector of values of length 1.

To print this quantity to the log file (with a descriptive column heading) the following commands could be included in an input script:

```
compute gauss all pair gauss
variable occ equal c_gauss[1]
thermo_style custom step temp epair v_occ
```

4.116.5 Restrictions

The *gauss* and *gauss/cut* styles are part of the EXTRA-PAIR package. They are only enabled if LAMMPS is build with that package. See the *Build package* page for more info.

Changed in version 28Mar2023.

Prior to this version, the *gauss* pair style did not apply *special_bonds* factors.

4.116.6 Related commands

pair_coeff, *pair_style coul/diel*

4.116.7 Default

none

(Lenart) Lenart , Jusufi, and Panagiotopoulos, J Chem Phys, 126, 044509 (2007).

(Jusufi) Jusufi, Hynninen, and Panagiotopoulos, J Phys Chem B, 112, 13783 (2008).

4.117 pair_style gayberne command

Accelerator Variants: *gayberne/gpu*, *gayberne/intel*, *gayberne/omp*

4.117.1 Syntax

```
pair_style gayberne gamma upsilon mu cutoff
```

- gamma = shift for potential minimum (typically 1)
- upsilon = exponent for eta orientation-dependent energy function
- mu = exponent for chi orientation-dependent energy function
- cutoff = global cutoff for interactions (distance units)

4.117.2 Examples

```
pair_style gayberne 1.0 1.0 1.0 10.0
pair_coeff * * 1.0 1.7 1.7 3.4 3.4 1.0 1.0 1.0
```

4.117.3 Description

The *gayberne* styles compute a Gay-Berne anisotropic LJ interaction *(Berardi)* between pairs of ellipsoidal particles or an ellipsoidal and spherical particle via the formulas

$$
\begin{aligned}
U(\mathbf{A}_1, \mathbf{A}_2, \mathbf{r}_{12}) =& U_r(\mathbf{A}_1, \mathbf{A}_2, \mathbf{r}_{12}, \gamma) \cdot \eta_{12}(\mathbf{A}_1, \mathbf{A}_2, \upsilon) \cdot \chi_{12}(\mathbf{A}_1, \mathbf{A}_2, \mathbf{r}_{12}, \mu) \\
U_r =& 4\varepsilon(\rho^{12} - \rho^6) \\
\rho =& \frac{\sigma}{h_{12} + \gamma\sigma}
\end{aligned}
$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are the transformation matrices from the simulation box frame to the body frame and r_{12} is the center to center vector between the particles. U_r controls the shifted distance dependent interaction based on the distance of closest approach of the two particles (h_{12}) and the user-specified shift parameter γ. When both particles are spherical, the formula reduces to the usual Lennard-Jones interaction (see details below for when Gay-Berne treats a particle as "spherical").

For large uniform molecules it has been shown that the energy parameters are approximately representable in terms of local contact curvatures *(Everaers)*:

$$
\varepsilon_a = \sigma \cdot \frac{a}{b \cdot c}; \varepsilon_b = \sigma \cdot \frac{b}{a \cdot c}; \varepsilon_c = \sigma \cdot \frac{c}{a \cdot b}
$$

The variable names utilized as potential parameters are for the most part taken from *(Everaers)* in order to be consistent with the *RE-squared pair potential*. Details on the upsilon and mu parameters are given here.

More details of the Gay-Berne formulation are given in the references listed below and in this supplementary document.

Use of this pair style requires the NVE, NVT, or NPT fixes with the *asphere* extension (e.g. *fix nve/asphere*) in order to integrate particle rotation. Additionally, *atom_style ellipsoid* should be used since it defines the rotational state and the size and shape of each ellipsoidal particle.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε = well depth (energy units)
- σ = minimum effective particle radii (distance units)
- $\varepsilon_{i,a}$ = relative well depth of type I for side-to-side interactions
- $\varepsilon_{i,b}$ = relative well depth of type I for face-to-face interactions
- $\varepsilon_{i,c}$ = relative well depth of type I for end-to-end interactions
- $\varepsilon_{j,a}$ = relative well depth of type J for side-to-side interactions
- $\varepsilon_{j,b}$ = relative well depth of type J for face-to-face interactions
- $\varepsilon_{j,c}$ = relative well depth of type J for end-to-end interactions
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used.

It is typical with the Gay-Berne potential to define σ as the minimum of the 3 shape diameters of the particles involved in an I,I interaction, though this is not required. Note that this is a different meaning for σ than the *pair_style resquared* potential uses.

The ε_i and ε_j coefficients are actually defined for atom types, not for pairs of atom types. Thus, in a series of pair_coeff commands, they only need to be specified once for each atom type.

Specifically, if any of $\varepsilon_{i,a}$, $\varepsilon_{i,b}$, $\varepsilon_{i,c}$ are non-zero, the three values are assigned to atom type I. If all the ε_i values are zero, they are ignored. If any of $\varepsilon_{j,a}$, $\varepsilon_{j,b}$, $\varepsilon_{j,c}$ are non-zero, the three values are assigned to atom type J. If all three epsilon_j values are zero, they are ignored. Thus the typical way to define the ε_i and ε_j coefficients is to list their values in "pair_coeff I J" commands when I = J, but set them to 0.0 when I != J. If you do list them when I != J, you should ensure they are consistent with their values in other pair_coeff commands, since only the last setting will be in effect.

Note that if this potential is being used as a sub-style of *pair_style hybrid*, and there is no "pair_coeff I I" setting made for Gay-Berne for a particular type I (because I-I interactions are computed by another hybrid pair potential), then you still need to ensure the ε a,b,c coefficients are assigned to that type. e.g. in a "pair_coeff I J" command.

Note

If the $\varepsilon_a = \varepsilon_b = \varepsilon_c$ for an atom type, and if the shape of the particle itself is spherical, meaning its 3 shape parameters are all the same, then the particle is treated as an LJ sphere by the Gay-Berne potential. This is significant because if two LJ spheres interact, then the simple Lennard-Jones formula is used to compute their interaction energy/force using the specified epsilon and sigma as the standard LJ parameters. This is much cheaper to compute than the full Gay-Berne formula. To treat the particle as a LJ sphere with sigma = D, you should normally set $\varepsilon_a = \varepsilon_b = \varepsilon_c = 1.0$, set the pair_coeff $\sigma = D$, and also set the 3 shape parameters for the particle to D. The one exception is that if the 3 shape parameters are set to 0.0, which is a valid way in LAMMPS to specify a point particle, then the Gay-Berne potential will treat that as shape parameters of 1.0 (i.e. a LJ particle with $\sigma = 1$), since it requires finite-size particles. In this case you should still set the pair_coeff σ to 1.0 as well.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.117.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction, but only for sphere-sphere interactions. There is no shifting performed for ellipsoidal interactions due to the anisotropic dependence of the interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.117.5 Restrictions

The *gayberne* style is part of the ASPHERE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles require that atoms store torque and a quaternion to represent their orientation, as defined by the *atom_style*. It also require they store a per-type *shape*. The particles cannot store a per-particle diameter.

This pair style requires that atoms be ellipsoids as defined by the *atom_style ellipsoid* command.

Particles acted on by the potential can be finite-size aspherical or spherical particles, or point particles. Spherical particles have all 3 of their shape parameters equal to each other. Point particles have all 3 of their shape parameters equal to 0.0.

The Gay-Berne potential does not become isotropic as r increases *(Everaers)*. The distance-of-closest-approach approximation used by LAMMPS becomes less accurate when high-aspect ratio ellipsoids are used.

4.117.6 Related commands

pair_coeff, *fix nve/asphere*, *compute temp/asphere*, *pair_style resquared*

4.117.7 Default

none

(Everaers) Everaers and Ejtehadi, Phys Rev E, 67, 041710 (2003).

(Berardi) Berardi, Fava, Zannoni, Chem Phys Lett, 297, 8-14 (1998). Berardi, Muccioli, Zannoni, J Chem Phys, 128, 024905 (2008).

(Perram) Perram and Rasmussen, Phys Rev E, 54, 6565-6572 (1996).

(Allen) Allen and Germano, Mol Phys 104, 3225-3235 (2006).

4.118 pair_style gran/hooke command

Accelerator Variants: *gran/hooke/omp*

4.119 pair_style gran/hooke/history command

Accelerator Variants: *gran/hooke/history/omp*, *gran/hooke/history/kk*

4.120 pair_style gran/hertz/history command

Accelerator Variants: *gran/hertz/history/omp*

4.120.1 Syntax

```
pair_style style Kn Kt gamma_n gamma_t xmu dampflag keyword
```

- style = *gran/hooke* or *gran/hooke/history* or *gran/hertz/history*
- Kn = elastic constant for normal particle repulsion (force/distance units or pressure units - see discussion below)
- Kt = elastic constant for tangential contact (force/distance units or pressure units - see discussion below)
- gamma_n = damping coefficient for collisions in normal direction (1/time units or 1/time-distance units - see discussion below)
- gamma_t = damping coefficient for collisions in tangential direction (1/time units or 1/time-distance units - see discussion below)
- xmu = static yield criterion (unitless value between 0.0 and 1.0e4)
- dampflag = 0 or 1 if tangential damping force is excluded or included
- keyword = *limit_damping*

limit_damping value = none
  limit damping to prevent attractive interaction

Note

Versions of LAMMPS before 9Jan09 had different style names for granular force fields. This is to emphasize the fact that the Hertzian equation has changed to model polydispersity more accurately. A side effect of the change is that the Kn, Kt, gamma_n, and gamma_t coefficients in the pair_style command must be specified with different values in order to reproduce calculations made with earlier versions of LAMMPS, even for monodisperse systems. See the NOTE below for details.

4.120.2 Examples

```
pair_style gran/hooke/history 200000.0 NULL 50.0 NULL 0.5 1
pair_style gran/hooke 200000.0 70000.0 50.0 30.0 0.5 0
pair_style gran/hooke 200000.0 70000.0 50.0 30.0 0.5 0 limit_damping
```

4.120.3 Description

The *gran* styles use the following formulas for the frictional force between two granular particles, as described in *(Brilliantov)*, *(Silbert)*, and *(Zhang)*, when the distance r between two particles of radii Ri and Rj is less than their contact distance d = Ri + Rj. There is no force between the particles when r > d.

The two Hookean styles use this formula:

$$F_{hk} = (k_n \delta \mathbf{n}_{ij} - m_{eff} \gamma_n \mathbf{v}_n) - (k_t \Delta \mathbf{s}_t + m_{eff} \gamma_t \mathbf{v}_t)$$

The Hertzian style uses this formula:

$$F_{hz} = \sqrt{\delta}\sqrt{\frac{R_i R_j}{R_i + R_j}} F_{hk} = \sqrt{\delta}\sqrt{\frac{R_i R_j}{R_i + R_j}} \left[(k_n \delta \mathbf{n}_{ij} - m_{eff}\ \gamma_n \mathbf{v}_n) - (k_t \Delta \mathbf{s}_t + m_{eff}\ \gamma_t \mathbf{v}_t)\right]$$

In both equations the first parenthesized term is the normal force between the two particles and the second parenthesized term is the tangential force. The normal force has 2 terms, a contact force and a damping force. The tangential force also has 2 terms: a shear force and a damping force. The shear force is a "history" effect that accounts for the tangential displacement between the particles for the duration of the time they are in contact. This term is included in pair styles *hooke/history* and *hertz/history*, but is not included in pair style *hooke*. The tangential damping force term is included in all three pair styles if *dampflag* is set to 1; it is not included if *dampflag* is set to 0.

The other quantities in the equations are as follows:

- δ = d - r = overlap distance of 2 particles
- K_n = elastic constant for normal contact
- K_t = elastic constant for tangential contact
- γ_n = viscoelastic damping constant for normal contact
- γ_t = viscoelastic damping constant for tangential contact
- $m_{eff} = M_i M_j / (M_i + M_j)$ = effective mass of 2 particles of mass M_i and M_j
- $\Delta \mathbf{s}_t$ = tangential displacement vector between 2 particles which is truncated to satisfy a frictional yield criterion
- n_{ij} = unit vector along the line connecting the centers of the 2 particles

- V_n = normal component of the relative velocity of the 2 particles
- V_t = tangential component of the relative velocity of the 2 particles

The K_n, K_t, γ_n, and γ_t coefficients are specified as parameters to the pair_style command. If a NULL is used for K_t, then a default value is used where $K_t = 2/7K_n$. If a NULL is used for γ_t, then a default value is used where $\gamma_t = 1/2\gamma_n$.

The interpretation and units for these 4 coefficients are different in the Hookean versus Hertzian equations.

The Hookean model is one where the normal push-back force for two overlapping particles is a linear function of the overlap distance. Thus the specified K_n is in units of (force/distance). Note that this push-back force is independent of absolute particle size (in the monodisperse case) and of the relative sizes of the two particles (in the polydisperse case). This model also applies to the other terms in the force equation so that the specified γ_n is in units of (1/time), K_t is in units of (force/distance), and γ_t is in units of (1/time).

The Hertzian model is one where the normal push-back force for two overlapping particles is proportional to the area of overlap of the two particles, and is thus a non-linear function of overlap distance. Thus Kn has units of force per area and is thus specified in units of (pressure). The effects of absolute particle size (monodispersity) and relative size (polydispersity) are captured in the radii-dependent prefactors. When these prefactors are carried through to the other terms in the force equation it means that the specified γ_n is in units of (1/(time*distance)), K_t is in units of (pressure), and γ_t is in units of (1/(time*distance)).

Note that in the Hookean case, K_n can be thought of as a linear spring constant with units of force/distance. In the Hertzian case, K_n is like a non-linear spring constant with units of force/area or pressure, and as shown in the *(Zhang)* paper, $K_n = 4G/(3(1-\nu))$ where ν = the Poisson ratio, G = shear modulus = $E/(2(1+\nu))$, and E = Young's modulus. Similarly, $K_t = 4G/(2-\nu)$. (NOTE: in an earlier version of the manual, we incorrectly stated that $K_t = 8G/(2-\nu)$.)

Thus in the Hertzian case K_n and K_t can be set to values that corresponds to properties of the material being modeled. This is also true in the Hookean case, except that a spring constant must be chosen that is appropriate for the absolute size of particles in the model. Since relative particle sizes are not accounted for, the Hookean styles may not be a suitable model for polydisperse systems.

Note

In versions of LAMMPS before 9Jan09, the equation for Hertzian interactions did not include the $\sqrt{r_i r_j/(r_i+r_j)}$ term and thus was not as accurate for polydisperse systems. For monodisperse systems, $\sqrt{r_i r_j/(r_i+r_j)}$ is a constant factor that effectively scales all 4 coefficients: $K_n, K_t, \gamma_n, \gamma_t$. Thus you can set the values of these 4 coefficients appropriately in the current code to reproduce the results of a previous Hertzian monodisperse calculation. For example, for the common case of a monodisperse system with particles of diameter 1, all 4 of these coefficients should now be set 2x larger than they were previously.

Xmu is also specified in the pair_style command and is the upper limit of the tangential force through the Coulomb criterion Ft = xmu*Fn, where Ft and Fn are the total tangential and normal force components in the formulas above. Thus in the Hookean case, the tangential force between 2 particles grows according to a tangential spring and dash-pot model until Ft/Fn = xmu and is then held at Ft = Fn*xmu until the particles lose contact. In the Hertzian case, a similar analogy holds, though the spring is no longer linear.

Note

Normally, xmu should be specified as a fractional value between 0.0 and 1.0, however LAMMPS allows large values (up to 1.0e4) to allow for modeling of systems which can sustain very large tangential forces.

The effective mass *m_eff* is given by the formula above for two isolated particles. If either particle is part of a rigid body, its mass is replaced by the mass of the rigid body in the formula above. This is determined by searching for a *fix rigid* command (or its variants).

For granular styles there are no additional coefficients to set for each pair of atom types via the *pair_coeff* command. All settings are global and are made via the pair_style command. However you must still use the *pair_coeff* for all pairs of granular atom types. For example the command

```
pair_coeff * *
```

should be used if all atoms in the simulation interact via a granular potential (i.e. one of the pair styles above is used). If a granular potential is used as a sub-style of *pair_style hybrid*, then specific atom types can be used in the pair_coeff command to determine which atoms interact via a granular potential.

If two particles are moving away from each other while in contact, there is a possibility that the particles could experience an effective attractive force due to damping. If the *limit_damping* keyword is used, this option will zero out the normal component of the force if there is an effective attractive force.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.120.4 Mixing, shift, table, tail correction, restart, rRESPA info

The *pair_modify* mix, shift, table, and tail options are not relevant for granular pair styles.

These pair styles write their information to *binary restart files*, so a pair_style command does not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

The single() function of these pair styles returns 0.0 for the energy of a pairwise interaction, since energy is not conserved in these dissipative potentials. It also returns only the normal component of the pairwise interaction force. However, the single() function also calculates 10 extra pairwise quantities. The first 3 are the components of the tangential force between particles I and J, acting on particle I. The fourth is the magnitude of this tangential force. The next 3 (5-7) are the components of the relative velocity in the normal direction (along the line joining the 2 sphere centers). The last 3 (8-10) the components of the relative velocity in the tangential direction.

These extra quantities can be accessed by the *compute pair/local* command, as *p1*, *p2*, ..., *p10*.

4.120.5 Restrictions

All the granular pair styles are part of the GRANULAR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles require that atoms store torque and angular velocity (omega) as defined by the *atom_style*. They also require a per-particle radius is stored. The *sphere* atom style does all of this.

This pair style requires you to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

These pair styles will not restart exactly when using the *read_restart* command, though they should provide statistically similar results. This is because the forces they compute depend on atom velocities. See the *read_restart* command for more details.

Accumulated values for individual contacts are saved to to restart files but are not saved to data files. Therefore, forces may differ significantly when a system is reloaded using A *read_data* command.

4.120.6 Related commands

pair_coeff

4.120.7 Default

none

(Brilliantov) Brilliantov, Spahn, Hertzsch, Poschel, Phys Rev E, 53, p 5382-5392 (1996).

(Silbert) Silbert, Ertas, Grest, Halsey, Levine, Plimpton, Phys Rev E, 64, p 051302 (2001).

(Zhang) Zhang and Makse, Phys Rev E, 72, p 011301 (2005).

4.121 pair_style granular command

4.121.1 Syntax

```
pair_style granular cutoff
```

- cutoff = global cutoff (optional). See discussion below.

4.121.2 Examples

```
pair_style granular
pair_coeff * * hooke 1000.0 50.0 tangential linear_nohistory 1.0 0.4 damping mass_velocity

pair_style granular
pair_coeff * * hooke 1000.0 50.0 tangential linear_history 500.0 1.0 0.4 damping mass_velocity

pair_style granular
pair_coeff * * hertz 1000.0 50.0 tangential mindlin 1000.0 1.0 0.4 limit_damping
```

(continues on next page)

(continued from previous page)

```
pair_style granular
pair_coeff * * hertz/material 1e8 0.3 0.3 tangential mindlin_rescale NULL 1.0 0.4 damping tsuji

pair_style granular
pair_coeff 1 * jkr 1000.0 500.0 0.3 10 tangential mindlin 800.0 1.0 0.5 rolling sds 500.0 200.0 0.5 twisting␣
↪marshall
pair_coeff 2 2 hertz 200.0 100.0 tangential linear_history 300.0 1.0 0.1 rolling sds 200.0 100.0 0.1␣
↪twisting marshall

pair_style granular
pair_coeff 1 1 dmt 1000.0 50.0 0.3 0.0 tangential mindlin NULL 0.5 0.5 rolling sds 500.0 200.0 0.5␣
↪twisting marshall
pair_coeff 2 2 dmt 1000.0 50.0 0.3 10.0 tangential mindlin NULL 0.5 0.1 rolling sds 500.0 200.0 0.1␣
↪twisting marshall

pair_style granular
pair_coeff * * hertz 1000.0 50.0 tangential mindlin 1000.0 1.0 0.4 heat area 0.1
```

4.121.3 Description

The *granular* styles support a variety of options for the normal, tangential, rolling and twisting forces resulting from contact between two granular particles. This expands on the options offered by the *pair gran/** pair styles. The total computed forces and torques are the sum of various models selected for the normal, tangential, rolling and twisting modes of motion.

All model choices and parameters are entered in the *pair_coeff* command, as described below. Unlike e.g. *pair gran/hooke*, coefficient values are not global, but can be set to different values for different combinations of particle types, as determined by the *pair_coeff* command. If the contact model choice is the same for two particle types, the mixing for the cross-coefficients can be carried out automatically. This is shown in the last example, where model choices are the same for type 1 - type 1 as for type 2 - type2 interactions, but coefficients are different. In this case, the mixed coefficients for type 1 - type 2 interactions can be determined from mixing rules discussed below. For additional flexibility, coefficients as well as model forms can vary between particle types, as shown in the fourth example: type 1 - type 1 interactions are based on a Johnson-Kendall-Roberts normal contact model and 2-2 interactions are based on a DMT cohesive model (see below). In that example, 1-1 and 2-2 interactions have different model forms, in which case mixing of coefficients cannot be determined, so 1-2 interactions must be explicitly defined via the *pair_coeff 1 ** command, otherwise an error would result.

The first required keyword for the *pair_coeff* command is the normal contact model. Currently supported options for normal contact models and their required arguments are:

1. *hooke* : k_n, η_{n0} (or e)
2. *hertz* : k_n, η_{n0} (or e)
3. *hertz/material* : E, η_{n0} (or e), ν
4. *dmt* : E, η_{n0} (or e), ν, γ
5. *jkr* : E, η_{n0} (or e), ν, γ

Here, k_n is spring stiffness (with units that depend on model choice, see below); η_{n0} is a damping prefactor (or, in its place a coefficient of restitution e, depending on the choice of damping mode, see below); E is Young's modulus in units of *force/length*^2, i.e. *pressure*; ν is Poisson's ratio and γ is a surface energy density, in units of *energy/length*^2.

For the *hooke* model, the normal, elastic component of force acting on particle i due to contact with particle j is given by:

$$\mathbf{F}_{ne,Hooke} = k_n \delta_{ij} \mathbf{n}$$

Where $\delta_{ij} = R_i + R_j - \|\mathbf{r}_{ij}\|$ is the particle overlap, R_i, R_j are the particle radii, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ is the vector separating the two particle centers (note the i-j ordering so that $\mathbf{F}_{ne}$ is positive for repulsion), and $\mathbf{n} = \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}$. Therefore, for *hooke*, the units of the spring constant k_n are *force/distance*, or equivalently *mass/time^2*.

For the *hertz* model, the normal component of force is given by:

$$\mathbf{F}_{ne,Hertz} = k_n R_{eff}^{1/2} \delta_{ij}^{3/2} \mathbf{n}$$

Here, $R_{eff} = R = \frac{R_i R_j}{R_i + R_j}$ is the effective radius, denoted for simplicity as R from here on. For *hertz*, the units of the spring constant k_n are *force/length^2*, or equivalently *pressure*.

For the *hertz/material* model, the force is given by:

$$\mathbf{F}_{ne,Hertz/material} = \frac{4}{3} E_{eff} R^{1/2} \delta_{ij}^{3/2} \mathbf{n}$$

Here, $E_{eff} = E = \left(\frac{1-\nu_i^2}{E_i} + \frac{1-\nu_j^2}{E_j} \right)^{-1}$ is the effective Young's modulus, with ν_i, ν_j the Poisson ratios of the particles of types i and j. E_{eff} is denoted as E from here on. Note that if the elastic modulus and the shear modulus of the two particles are the same, the *hertz/material* model is equivalent to the *hertz* model with $k_n = 4/3E$

The *dmt* model corresponds to the *(Derjaguin-Muller-Toporov)* cohesive model, where the force is simply Hertz with an additional attractive cohesion term:

$$\mathbf{F}_{ne,dmt} = \left(\frac{4}{3} E R^{1/2} \delta_{ij}^{3/2} - 4\pi\gamma R \right) \mathbf{n}$$

The *jkr* model is the *(Johnson-Kendall-Roberts)* model, where the force is computed as:

$$\mathbf{F}_{ne,jkr} = \left(\frac{4Ea^3}{3R} - 2\pi a^2 \sqrt{\frac{4\gamma E}{\pi a}} \right) \mathbf{n}$$

Here, a is the radius of the contact zone, related to the overlap δ according to:

$$\delta = a^2/R - 2\sqrt{\pi\gamma a/E}$$

LAMMPS internally inverts the equation above to solve for a in terms of δ, then solves for the force in the previous equation. Additionally, note that the JKR model allows for a tensile force beyond contact (i.e. for $\delta < 0$), up to a maximum of $3\pi\gamma R$ (also known as the 'pull-off' force). Note that this is a hysteretic effect, where particles that are not contacting initially will not experience force until they come into contact $\delta \geq 0$; as they move apart and ($\delta < 0$), they experience a tensile force up to $3\pi\gamma R$, at which point they lose contact.

In addition, the normal force is augmented by a damping term of the following general form:

$$\mathbf{F}_{n,damp} = -\eta_n \mathbf{v}_{n,rel}$$

Here, $\mathbf{v}_{n,rel} = (\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{n}\, \mathbf{n}$ is the component of relative velocity along $\mathbf{n}$.

The optional *damping* keyword to the *pair_coeff* command followed by a keyword determines the model form of the damping factor η_n, and the interpretation of the η_{n0} or e coefficients specified as part of the normal contact model settings. The *damping* keyword and corresponding model form selection may be appended anywhere in the *pair coeff* command. Note that the choice of damping model affects both the normal and tangential damping (and depending on other settings, potentially also the twisting damping). The options for the damping model currently supported are:

1. *velocity*
2. *mass_velocity*
3. *viscoelastic*
4. *tsuji*
5. *coeff_restitution*

If the *damping* keyword is not specified, the *viscoelastic* model is used by default.

For *damping velocity*, the normal damping is simply equal to the user-specified damping coefficient in the *normal* model:

$$\eta_n = \eta_{n0}$$

Here, η_{n0} is the damping coefficient specified for the normal contact model, in units of *mass*/*time*.

For *damping mass_velocity*, the normal damping is given by:

$$\eta_n = \eta_{n0} m_{eff}$$

Here, η_{n0} is the damping coefficient specified for the normal contact model, in units of 1/*time* and $m_{eff} = m_i m_j/(m_i+m_j)$ is the effective mass. Use *damping mass_velocity* to reproduce the damping behavior of *pair gran/hooke/**.

The *damping viscoelastic* model is based on the viscoelastic treatment of *(Brilliantov et al)*, where the normal damping is given by:

$$\eta_n = \eta_{n0}\, a m_{eff}$$

Here, a is the contact radius, given by $a = \sqrt{R\delta}$ for all models except *jkr*, for which it is given implicitly according to $\delta = a^2/R - 2\sqrt{\pi\gamma a/E}$. For *damping viscoelastic*, η_{n0} is in units of 1/(*time***distance*).

The *tsuji* model is based on the work of *(Tsuji et al)*. Here, the damping coefficient specified as part of the normal model is interpreted as a restitution coefficient e. The damping constant η_n is given by:

$$\eta_n = \alpha (m_{eff} k_n)^{1/2}$$

For normal contact models based on material parameters, $k_n = 4/3Ea$. This damping model is not compatible with cohesive normal models such as *JKR* or *DMT*. The parameter α is related to the restitution coefficient e according to:

$$\alpha = 1.2728 - 4.2783e + 11.087e^2 - 22.348e^3 + 27.467e^4 - 18.022e^5 + 4.8218e^6$$

The dimensionless coefficient of restitution e specified as part of the normal contact model parameters should be between 0 and 1, but no error check is performed on this.

The *coeff_restitution* model is useful when a specific normal coefficient of restitution e is required. In these models, the normal coefficient of restitution e is specified as an input. Following the approach of *(Brilliantov et al)*, when using the *hooke* normal model, *coeff_restitution* calculates the damping coefficient as:

$$\eta_n = \sqrt{\frac{4 m_{eff} k_n}{1 + \left(\frac{\pi}{\log(e)}\right)^2}},$$

For any other normal model, e.g. the *hertz* and *hertz/material* models, the damping coefficient is:

$$\eta_n = -2\sqrt{\frac{5}{6}} \frac{\log(e)}{\sqrt{\pi^2 + (\log(e))^2}} (R_{eff}\delta_{ij})^{\frac{1}{4}} \sqrt{\frac{3}{2} k_n m_{eff}},$$

where $k_n = \frac{4}{3} E_{eff}$ for the *hertz/material* model. Since *coeff_restitution* accounts for the effective mass, effective radius, and pairwise overlaps (except when used with the *hooke* normal model) when calculating the damping coefficient, it

accurately reproduces the specified coefficient of restitution for both monodisperse and polydisperse particle pairs. This damping model is not compatible with cohesive normal models such as *JKR* or *DMT*.

The total normal force is computed as the sum of the elastic and damping components:

$$\mathbf{F}_n = \mathbf{F}_{ne} + \mathbf{F}_{n,damp}$$

The *pair_coeff* command also requires specification of the tangential contact model. The required keyword *tangential* is expected, followed by the model choice and associated parameters. Currently supported tangential model choices and their expected parameters are as follows:

1. *linear_nohistory* : $x_{\gamma,t}$, μ_s
2. *linear_history* : k_t, $x_{\gamma,t}$, μ_s
3. *mindlin* : k_t or NULL, $x_{\gamma,t}$, μ_s
4. *mindlin/force* : k_t or NULL, $x_{\gamma,t}$, μ_s
5. *mindlin_rescale* : k_t or NULL, $x_{\gamma,t}$, μ_s
6. *mindlin_rescale/force* : k_t or NULL, $x_{\gamma,t}$, μ_s

Here, $x_{\gamma,t}$ is a dimensionless multiplier for the normal damping η_n that determines the magnitude of the tangential damping, μ_t is the tangential (or sliding) friction coefficient, and k_t is the tangential stiffness coefficient.

For *tangential linear_nohistory*, a simple velocity-dependent Coulomb friction criterion is used, which mimics the behavior of the *pair gran/hooke* style. The tangential force $\mathbf{F}_t$ is given by:

$$\mathbf{F}_t = -\min(\mu_t F_{n0}, \|\mathbf{F}_{\mathrm{t,damp}}\|)\mathbf{t}$$

The tangential damping force $\mathbf{F}_{\mathrm{t,damp}}$ is given by:

$$\mathbf{F}_{\mathrm{t,damp}} = -\eta_t \mathbf{v}_{t,rel}$$

The tangential damping prefactor η_t is calculated by scaling the normal damping η_n (see above):

$$\eta_t = -x_{\gamma,t}\eta_n$$

The normal damping prefactor η_n is determined by the choice of the *damping* keyword, as discussed above. Thus, the *damping* keyword also affects the tangential damping. The parameter $x_{\gamma,t}$ is a scaling coefficient. Several works in the literature use $x_{\gamma,t} = 1$ (*Marshall*, *Tsuji et al*, *Silbert et al*). The relative tangential velocity at the point of contact is given by $\mathbf{v}_{t,rel} = \mathbf{v}_t - (R_i\Omega_i + R_j\Omega_j) \times \mathbf{n}$, where $\mathbf{v}_t = \mathbf{v}_r - \mathbf{v}_r \cdot \mathbf{n}\,\mathbf{n}$, $\mathbf{v}_r = \mathbf{v}_j - \mathbf{v}_i$. The direction of the applied force is $\mathbf{t} = \mathbf{v}_{\mathbf{t,rel}}/\|\mathbf{v}_{\mathbf{t,rel}}\|$.

The normal force value F_{n0} used to compute the critical force depends on the form of the contact model. For non-cohesive models (*hertz*, *hertz/material*, *hooke*), it is given by the magnitude of the normal force:

$$F_{n0} = \|\mathbf{F}_n\|$$

For cohesive models such as *jkr* and *dmt*, the critical force is adjusted so that the critical tangential force approaches $\mu_t F_{pulloff}$, see *Marshall*, equation 43, and *Thornton*. For both models, F_{n0} takes the form:

$$F_{n0} = \|\mathbf{F}_{ne} + 2F_{pulloff}\|$$

Where $F_{pulloff} = 3\pi\gamma R$ for *jkr*, and $F_{pulloff} = 4\pi\gamma R$ for *dmt*.

The remaining tangential options all use accumulated tangential displacement (i.e. contact history), except for the options *mindlin/force* and *mindlin_rescale/force*, that use accumulated tangential force instead, and are discussed further below. The accumulated tangential displacement is discussed in details below in the context of the *linear_history* option. The same treatment of the accumulated displacement applies to the other options as well.

For *tangential linear_history*, the tangential force is given by:

$$\mathbf{F}_t = -\min(\mu_t F_{n0}, \| - k_t \xi + \mathbf{F}_{\mathrm{t,damp}}\|)\mathbf{t}$$

Here, ξ is the tangential displacement accumulated during the entire duration of the contact:

$$\xi = \int_{t0}^{t} \mathbf{v}_{t,rel}(\tau)\mathrm{d}\tau$$

This accumulated tangential displacement must be adjusted to account for changes in the frame of reference of the contacting pair of particles during contact. This occurs due to the overall motion of the contacting particles in a rigid-body-like fashion during the duration of the contact. There are two modes of motion that are relevant: the 'tumbling' rotation of the contacting pair, which changes the orientation of the plane in which tangential displacement occurs; and 'spinning' rotation of the contacting pair about the vector connecting their centers of mass (**n**). Corrections due to the former mode of motion are made by rotating the accumulated displacement into the plane that is tangential to the contact vector at each step, or equivalently removing any component of the tangential displacement that lies along **n**, and rescaling to preserve the magnitude. This follows the discussion in *Luding*, see equation 17 and relevant discussion in that work:

$$\xi = \left(\xi' - (\mathbf{n}\cdot\xi')\mathbf{n}\right)\frac{\|\xi'\|}{\|\xi' - (\mathbf{n}\cdot\xi')\mathbf{n}\|}$$

Here, ξ' is the accumulated displacement prior to the current time step and ξ is the corrected displacement. Corrections to the displacement due to the second mode of motion described above (rotations about **n**) are not currently implemented, but are expected to be minor for most simulations.

Furthermore, when the tangential force exceeds the critical force, the tangential displacement is re-scaled to match the value for the critical force (see *Luding*, equation 20 and related discussion):

$$\xi = -\frac{1}{k_t}\left(\mu_t F_{n0}\mathbf{t} - \mathbf{F}_{t,damp}\right)$$

The tangential force is added to the total normal force (elastic plus damping) to produce the total force on the particle. The tangential force also acts at the contact point (defined as the center of the overlap region) to induce a torque on each particle according to:

$$\tau_i = -(R_i - 0.5\delta)\mathbf{n}\times\mathbf{F}_t$$

$$\tau_j = -(R_j - 0.5\delta)\mathbf{n}\times\mathbf{F}_t$$

For *tangential mindlin*, the *Mindlin* no-slip solution is used which differs from the *linear_history* option by an additional factor of a, the radius of the contact region. The tangential force is given by:

$$\mathbf{F}_t = -\min(\mu_t F_{n0}, \| - k_t a\xi + \mathbf{F}_{\mathrm{t,damp}}\|)\mathbf{t}$$

Here, a is the radius of the contact region, given by $a = \sqrt{R\delta}$ for all normal contact models, except for *jkr*, where it is given implicitly by $\delta = a^2/R - 2\sqrt{\pi\gamma a/E}$, see discussion above. To match the Mindlin solution, one should set $k_t = 8G_{eff}$, where G_{eff} is the effective shear modulus given by:

$$G_{eff} = \left(\frac{2-\nu_i}{G_i} + \frac{2-\nu_j}{G_j}\right)^{-1}$$

where G_i is the shear modulus of a particle of type i, related to Young's modulus E_i and Poisson's ratio ν_i by $G_i = E_i/(2(1+\nu_i))$. This can also be achieved by specifying *NULL* for k_t, in which case a normal contact model that

specifies material parameters E_i and ν_i is required (e.g. *hertz/material*, *dmt* or *jkr*). In this case, mixing of the shear modulus for different particle types i and j is done according to the formula above.

Note

The radius of the contact region a depends on the normal overlap. As a result, the tangential force for *mindlin* can change due to a variation in normal overlap, even with no change in tangential displacement.

For *tangential mindlin/force*, the accumulated elastic tangential force characterizes the contact history, instead of the accumulated tangential displacement. This prevents the dependence of the tangential force on the normal overlap as noted above. The tangential force is given by:

$$\mathbf{F}_t = -\min(\mu_t F_{n0}, \|\mathbf{F}_{te} + \mathbf{F}_{\mathrm{t,damp}}\|)\mathbf{t}$$

The increment of the elastic component of the tangential force $\mathbf{F}_{te}$ is given by:

$$\mathrm{d}\mathbf{F}_{te} = -k_t a \mathbf{v}_{t,rel} \mathrm{d}\tau$$

The changes in frame of reference of the contacting pair of particles during contact are accounted for by the same formula as above, replacing the accumulated tangential displacement ξ, by the accumulated tangential elastic force F_{te}. When the tangential force exceeds the critical force, the tangential force is directly re-scaled to match the value for the critical force:

$$\mathbf{F}_{te} = -\mu_t F_{n0}\mathbf{t} + \mathbf{F}_{t,damp}$$

The same rules as those described for *mindlin* apply regarding the tangential stiffness and mixing of the shear modulus for different particle types.

The *mindlin_rescale* option uses the same form as *mindlin*, but the magnitude of the tangential displacement is re-scaled as the contact unloads, i.e. if $a < a_{t_{n-1}}$:

$$\xi = \xi_{\mathbf{t_{n-1}}} \frac{a}{a_{t_{n-1}}}$$

Here, t_{n-1} indicates the value at the previous time step. This rescaling accounts for the fact that a decrease in the contact area upon unloading leads to the contact being unable to support the previous tangential loading, and spurious energy is created without the rescaling above (*Walton*).

Note

For *mindlin*, a decrease in the tangential force already occurs as the contact unloads, due to the dependence of the tangential force on the normal force described above. By re-scaling ξ, *mindlin_rescale* effectively re-scales the tangential force twice, i.e., proportionally to a^2. This peculiar behavior results from use of the accumulated tangential displacement to characterize the contact history. Although *mindlin_rescale* remains available for historic reasons and backward compatibility purposes, it should be avoided in favor of *mindlin_rescale/force*.

The *mindlin_rescale/force* option uses the same form as *mindlin/force*, but the magnitude of the tangential elastic force is re-scaled as the contact unloads, i.e. if $a < a_{t_{n-1}}$:

$$\mathbf{F}_{te} = \mathbf{F}_{te,t_{n-1}} \frac{a}{a_{t_{n-1}}}$$

This approach provides a better approximation of the *Mindlin-Deresiewicz* laws and is more consistent than *mindlin_rescale*. See discussions in *Thornton et al, 2013*, particularly equation 18(b) of that work and associated discussion, and *Agnolin and Roux, 2007*, particularly Appendix A.

The optional *rolling* keyword enables rolling friction, which resists pure rolling motion of particles. The options currently supported are:

1. *none*
2. *sds* : k_{roll}, γ_{roll}, μ_{roll}

If the *rolling* keyword is not specified, the model defaults to *none*.

For *rolling sds*, rolling friction is computed via a spring-dashpot-slider, using a 'pseudo-force' formulation, as detailed by *Luding*. Unlike the formulation in *Marshall*, this allows for the required adjustment of rolling displacement due to changes in the frame of reference of the contacting pair. The rolling pseudo-force is computed analogously to the tangential force:

$$\mathbf{F}_{roll,0} = k_{roll}\xi_{roll} - \gamma_{roll}\mathbf{v}_{roll}$$

Here, $\mathbf{v}_{roll} = -R(\Omega_i - \Omega_j) \times \mathbf{n}$ is the relative rolling velocity, as given in *Wang et al* and *Luding*. This differs from the expressions given by *Kuhn and Bagi* and used in *Marshall*; see *Wang et al* for details. The rolling displacement is given by:

$$\xi_{roll} = \int_{t_0}^{t} \mathbf{v}_{roll}(\tau)\mathrm{d}\tau$$

A Coulomb friction criterion truncates the rolling pseudo-force if it exceeds a critical value:

$$\mathbf{F}_{roll} = \min(\mu_{roll}F_{n,0}, \|\mathbf{F}_{roll,0}\|)\mathbf{k}$$

Here, $\mathbf{k} = \mathbf{v}_{roll}/\|\mathbf{v}_{roll}\|$ is the direction of the pseudo-force. As with tangential displacement, the rolling displacement is rescaled when the critical force is exceeded, so that the spring length corresponds the critical force. Additionally, the displacement is adjusted to account for rotations of the frame of reference of the two contacting particles in a manner analogous to the tangential displacement.

The rolling pseudo-force does not contribute to the total force on either particle (hence 'pseudo'), but acts only to induce an equal and opposite torque on each particle, according to:

$$\tau_{roll,i} = R\mathbf{n} \times \mathbf{F}_{roll}$$

$$\tau_{roll,j} = -\tau_{roll,i}$$

The optional *twisting* keyword enables twisting friction, which resists rotation of two contacting particles about the vector **n** that connects their centers. The options currently supported are:

1. *none*
2. *sds* : k_{twist}, γ_{twist}, μ_{twist}
3. *marshall*

If the *twisting* keyword is not specified, the model defaults to *none*.

For both *twisting sds* and *twisting marshall*, a history-dependent spring-dashpot-slider is used to compute the twisting torque. Because twisting displacement is a scalar, there is no need to adjust for changes in the frame of reference due to rotations of the particle pair. The formulation in *Marshall* therefore provides the most straightforward treatment:

$$\tau_{twist,0} = -k_{twist}\xi_{twist} - \gamma_{twist}\Omega_{twist}$$

Here $\xi_{twist} = \int_{t_0}^{t} \Omega_{twist}(\tau)\mathrm{d}\tau$ is the twisting angular displacement, and $\Omega_{twist} = (\blacksquare_i - \blacksquare_j) \cdot \mathbf{n}$ is the relative twisting angular velocity. The torque is then truncated according to:

$$\tau_{twist} = \min(\mu_{twist} F_{n,0}, \tau_{twist,0})$$

Similar to the sliding and rolling displacement, the angular displacement is rescaled so that it corresponds to the critical value if the twisting torque exceeds this critical value:

$$\xi_{twist} = \frac{1}{k_{twist}}(\mu_{twist} F_{n,0} sgn(\Omega_{twist}) - \gamma_{twist}\Omega_{twist})$$

For *twisting sds*, the coefficients k_{twist}, γ_{twist} and μ_{twist} are simply the user input parameters that follow the *twisting sds* keywords in the *pair_coeff* command.

For *twisting_marshall*, the coefficients are expressed in terms of sliding friction coefficients, as discussed in *Marshall* (see equations 32 and 33 of that work):

$$k_{twist} = 0.5 k_t a^2$$

$$\eta_{twist} = 0.5 \eta_t a^2$$

$$\mu_{twist} = \frac{2}{3} a \mu_t$$

Finally, the twisting torque on each particle is given by:

$$\tau_{twist,i} = \tau_{twist} \mathbf{n}$$

$$\tau_{twist,j} = -\tau_{twist,i}$$

If two particles are moving away from each other while in contact, there is a possibility that the particles could experience an effective attractive force due to damping. If the optional *limit_damping* keyword is used, this option will zero out the normal component of the force if there is an effective attractive force. This keyword cannot be used with the JKR or DMT models.

The optional *heat* keyword enables heat conduction. The options currently supported are:

1. *none*
2. *radius* : k_s
3. *area* : h_s

If the *heat* keyword is not specified, the model defaults to *none*.

For *heat radius*, the heat Q conducted between two particles is given by

$$Q = 2 k_s a \Delta T$$

where ΔT is the difference in the two particles' temperature, k_s is a non-negative numeric value for the conductivity (in units of power/(length*temperature)), and a is the radius of the contact and depends on the normal force model. This is the model proposed by *Vargas and McCarthy*.

For *heat area*, the heat Q conducted between two particles is given by

$$Q = h_s A \Delta T$$

where ΔT is the difference in the two particles' temperature, h_s is a non-negative numeric value for the heat transfer coefficient (in units of power/(area*temperature)), and $A = \pi a^2$ is the area of the contact and depends on the normal force model.

Note that the option *none* must either be used in all or none of of the *pair_coeff* calls. See *fix heat/flow* and *fix property/atom* for more information on this option.

The *granular* pair style can reproduce the behavior of the *pair gran/** styles with the appropriate settings (some very minor differences can be expected due to corrections in displacement history frame-of-reference, and the application of the torque at the center of the contact rather than at each particle). The first example above is equivalent to *pair gran/hooke 1000.0 NULL 50.0 50.0 0.4 1*. The second example is equivalent to *pair gran/hooke/history 1000.0 500.0 50.0 50.0 0.4 1*. The third example is equivalent to *pair gran/hertz/history 1000.0 500.0 50.0 50.0 0.4 1 limit_damping*.

LAMMPS automatically sets pairwise cutoff values for *pair_style granular* based on particle radii (and in the case of *jkr* pull-off distances). In the vast majority of situations, this is adequate. However, a cutoff value can optionally be appended to the *pair_style granular* command to specify a global cutoff (i.e. a cutoff for all atom types). Additionally, the optional *cutoff* keyword can be passed to the *pair_coeff* command, followed by a cutoff value. This will set a pairwise cutoff for the atom types in the *pair_coeff* command. These options may be useful in some rare cases where the automatic cutoff determination is not sufficient, e.g. if particle diameters are being modified via the *fix adapt* command. In that case, the global cutoff specified as part of the *pair_style granular* command is applied to all atom types, unless it is overridden for a given atom type combination by the *cutoff* value specified in the *pair coeff* command. If *cutoff* is only specified in the *pair coeff* command and no global cutoff is appended to the *pair_style granular* command, then LAMMPS will use that cutoff for the specified atom type combination, and automatically set pairwise cutoffs for the remaining atom types.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.121.4 Mixing, shift, table, tail correction, restart, rRESPA info

The *pair_modify* mix, shift, table, and tail options are not relevant for granular pair styles.

Mixing of coefficients is carried out using geometric averaging for most quantities, e.g. if friction coefficient for type 1-type 1 interactions is set to μ_1, and friction coefficient for type 2-type 2 interactions is set to μ_2, the friction coefficient for type1-type2 interactions is computed as $\sqrt{\mu_1 \mu_2}$ (unless explicitly specified to a different value by a *pair_coeff 1 2 ...* command). The exception to this is elastic modulus, only applicable to *hertz/material*, *dmt* and *jkr* normal contact models. In that case, the effective elastic modulus is computed as:

$$E_{eff,ij} = \left(\frac{1 - \nu_i^2}{E_i} + \frac{1 - \nu_j^2}{E_j} \right)^{-1}$$

If the *i-j* coefficients E_{ij} and ν_{ij} are explicitly specified, the effective modulus is computed as:

$$E_{eff,ij} = \left(\frac{1 - \nu_{ij}^2}{E_{ij}} + \frac{1 - \nu_{ij}^2}{E_{ij}} \right)^{-1}$$

or

$$E_{eff,ij} = \frac{E_{ij}}{2(1 - \nu_{ij}^2)}$$

These pair styles write their information to *binary restart files*, so a pair_style command does not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

The single() function of these pair styles returns 0.0 for the energy of a pairwise interaction, since energy is not conserved in these dissipative potentials. It also returns only the normal component of the pairwise interaction force. However, the single() function also calculates 13 extra pairwise quantities. The first 3 are the components of the tangential force between particles I and J, acting on particle I. The fourth is the magnitude of this tangential force. The next 3 (5-7) are the components of the rolling torque acting on particle I. The next entry (8) is the magnitude of the rolling torque. The next entry (9) is the magnitude of the twisting torque acting about the vector connecting the two particle centers. The next 3 (10-12) are the components of the vector connecting the centers of the two particles (x_I - x_J). The last quantity (13) is the heat flow between the two particles, set to 0 if no heat model is active.

These extra quantities can be accessed by the *compute pair/local* command, as *p1*, *p2*, ..., *p12*.

4.121.5 Restrictions

This pair style is part of the GRANULAR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires that atoms store per-particle radius, torque, and angular velocity (omega) as defined by the *atom_style sphere*.

This pair style requires you to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

This pair style will not restart exactly when using the *read_restart* command, though it should provide statistically similar results. This is because the forces it computes depend on atom velocities and the atom velocities have been propagated half a timestep between the force computation and when the restart is written, due to using Velocity Verlet time integration. See the *read_restart* command for more details.

Accumulated values for individual contacts are saved to restart files but are not saved to data files. Therefore, forces may differ significantly when a system is reloaded using the *read_data* command.

4.121.6 Related commands

pair_coeff *pair gran/**

4.121.7 Default

For the *pair_coeff* settings: *damping viscoelastic*, *rolling none*, *twisting none*.

4.121.8 References

(Brilliantov et al, 1996) Brilliantov, N. V., Spahn, F., Hertzsch, J. M., & Poschel, T. (1996). Model for collisions in granular gases. Physical review E, 53(5), 5382.

(Tsuji et al, 1992) Tsuji, Y., Tanaka, T., & Ishida, T. (1992). Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe. Powder technology, 71(3), 239-250.

(Johnson et al, 1971) Johnson, K. L., Kendall, K., & Roberts, A. D. (1971). Surface energy and the contact of elastic solids. Proc. R. Soc. Lond. A, 324(1558), 301-313.

Derjaguin et al, 1975) Derjaguin, B. V., Muller, V. M., & Toporov, Y. P. (1975). Effect of contact deformations on the adhesion of particles. Journal of Colloid and interface science, 53(2), 314-326.

(Luding, 2008) Luding, S. (2008). Cohesive, frictional powders: contact models for tension. Granular matter, 10(4), 235.

(Marshall, 2009) Marshall, J. S. (2009). Discrete-element modeling of particulate aerosol flows. Journal of Computational Physics, 228(5), 1541-1561.

(Silbert, 2001) Silbert, L. E., Ertas, D., Grest, G. S., Halsey, T. C., Levine, D., & Plimpton, S. J. (2001). Granular flow down an inclined plane: Bagnold scaling and rheology. Physical Review E, 64(5), 051302.

(Kuhn and Bagi, 2005) Kuhn, M. R., & Bagi, K. (2004). Contact rolling and deformation in granular media. International journal of solids and structures, 41(21), 5793-5820.

(Wang et al, 2015) Wang, Y., Alonso-Marroquin, F., & Guo, W. W. (2015). Rolling and sliding in 3-D discrete element models. Particuology, 23, 49-55.

(Thornton, 1991) Thornton, C. (1991). Interparticle sliding in the presence of adhesion. J. Phys. D: Appl. Phys. 24 1942

(Mindlin, 1949) Mindlin, R. D. (1949). Compliance of elastic bodies in contact. J. Appl. Mech., ASME 16, 259-268.

(Thornton et al, 2013) Thornton, C., Cummins, S. J., & Cleary, P. W. (2013). An investigation of the comparative behavior of alternative contact force models during inelastic collisions. Powder Technology, 233, 30-46.

(Otis R. Walton) Walton, O.R., Personal Communication

(Mindlin and Deresiewicz, 1953) Mindlin, R.D., & Deresiewicz, H (1953). Elastic Spheres in Contact under Varying Oblique Force. J. Appl. Mech., ASME 20, 327-344.

(Agnolin and Roux 2007) Agnolin, I. & Roux, J-N. (2007). Internal states of model isotropic granular packings. I. Assembling process, geometry, and contact networks. Phys. Rev. E, 76, 061302.

(Vargas and McCarthy 2001) Vargas, W.L. and McCarthy, J.J. (2001). Heat conduction in granular materials. AIChE Journal, 47(5), 1052-1059.

4.122 pair_style lj/gromacs command

Accelerator Variants: *lj/gromacs/gpu*, *lj/gromacs/kk*, *lj/gromacs/omp*

4.123 pair_style lj/gromacs/coul/gromacs command

Accelerator Variants: *lj/gromacs/coul/gromacs/kk*, *lj/gromacs/coul/gromacs/omp*

4.123.1 Syntax

```
pair_style style args
```

- style = *lj/gromacs* or *lj/gromacs/coul/gromacs*
- args = list of arguments for a particular style

lj/gromacs args = inner outer
  inner, outer = global switching cutoffs for Lennard Jones
lj/gromacs/coul/gromacs args = inner outer (inner2) (outer2)
  inner, outer = global switching cutoffs for Lennard Jones (and Coulombic if only 2 args)
  inner2, outer2 = global switching cutoffs for Coulombic (optional)

4.123.2 Examples

```
pair_style lj/gromacs 9.0 12.0
pair_coeff * * 100.0 2.0
pair_coeff 2 2 100.0 2.0 8.0 10.0

pair_style lj/gromacs/coul/gromacs 9.0 12.0
pair_style lj/gromacs/coul/gromacs 8.0 10.0 7.0 9.0
pair_coeff * * 100.0 2.0
```

4.123.3 Description

The *lj/gromacs* styles compute shifted LJ and Coulombic interactions with an additional switching function S(r) that ramps the energy and force smoothly to zero between an inner and outer cutoff. It is a commonly used potential in the GROMACS MD code and for the coarse-grained models of *(Marrink)*.

$$
\begin{aligned}
E_{LJ} &= 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] + S_{LJ}(r) \qquad r < r_c \\
E_C &= \frac{C q_i q_j}{\varepsilon r} + S_C(r) \qquad r < r_c \\
S(r) &= C \qquad r < r_1 \\
S(r) &= \frac{A}{3}(r - r_1)^3 + \frac{B}{4}(r - r_1)^4 + C \qquad r_1 < r < r_c \\
A &= (-3E'(r_c) + (r_c - r_1)E''(r_c))/(r_c - r_1)^2 \\
B &= (2E'(r_c) - (r_c - r_1)E''(r_c))/(r_c - r_1)^3 \\
C &= -E(r_c) + \frac{1}{2}(r_c - r_1)E'(r_c) - \frac{1}{12}(r_c - r_1)^2 E''(r_c)
\end{aligned}
$$

r_1 is the inner cutoff; r_c is the outer cutoff. The coefficients A, B, and C are computed by LAMMPS to perform the shifting and smoothing. The function S(r) is actually applied once to each term of the LJ formula and once to the Coulombic formula, so there are 2 or 3 sets of A,B,C coefficients depending on which pair_style is used. The boundary conditions applied to the smoothing function are as follows: $S'(r_1) = S''(r_1) = 0, S(r_c) = -E(r_c), S'(r_c) = -E'(r_c)$, and $S''(r_c) = -E''(r_c)$, where E(r) is the corresponding term in the LJ or Coulombic potential energy function. Single and double primes denote first and second derivatives with respect to r, respectively.

The inner and outer cutoff for the LJ and Coulombic terms can be the same or different depending on whether 2 or 4 arguments are used in the pair_style command. The inner LJ cutoff must be > 0, but the inner Coulombic cutoff can be >= 0.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- inner (distance units)
- outer (distance units)

Note that sigma is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum at $2^{1/6}\sigma$.

The last 2 coefficients are optional inner and outer cutoffs for style *lj/gromacs*. If not specified, the global *inner* and *outer* values are used.

The last 2 coefficients cannot be used with style *lj/gromacs/coul/gromacs* because this force field does not allow varying cutoffs for individual atom pairs; all pairs use the global cutoff(s) specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.123.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

None of the GROMACS pair styles support the *pair_modify* shift option, since the Lennard-Jones portion of the pair interaction is already smoothed to 0.0 at the cutoff.

The *pair_modify* table option is not relevant for this pair style.

None of the GROMACS pair styles support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since there are no corrections for a potential that goes to 0.0 at the cutoff.

All of the GROMACS pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

All of the GROMACS pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.123.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.123.6 Related commands

pair_coeff

4.123.7 Default

none

(Marrink) Marrink, de Vries, Mark, J Phys Chem B, 108, 750-760 (2004).

4.124 pair_style gw command

4.125 pair_style gw/zbl command

4.125.1 Syntax

```
pair_style style
```

- style = *gw* or *gw/zbl*

4.125.2 Examples

```
pair_style gw
pair_coeff * * SiC.gw Si C C

pair_style gw/zbl
pair_coeff * * SiC.gw.zbl C Si
```

4.125.3 Description

The *gw* style computes a 3-body *Gao-Weber* potential; similarly *gw/zbl* combines this potential with a modified repulsive ZBL core function in a similar fashion as implemented in the *tersoff/zbl* pair style.

Unfortunately the author of this contributed code has not been able to submit a suitable documentation explaining the details of the potentials. The LAMMPS developers thus have finally decided to release the code anyway with only the technical explanations. For details of the model and the parameters, please refer to the linked publication.

Only a single pair_coeff command is used with the *gw* and *gw/zbl* styles which specifies a Gao-Weber potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of GW elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.gw has Gao-Weber values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.gw Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the GW file. The final C argument maps LAMMPS atom type 4 to the C element in the GW file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *gw* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Gao-Weber files in the *potentials* directory of the LAMMPS distribution have a ".gw" suffix. Gao-Weber with ZBL files have a ".gz.zbl" suffix. The structure of the potential files is similar to other many-body potentials supported by LAMMPS. You have to refer to the comments in the files and the literature to learn more details.

4.125.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.125.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The Gao-Weber potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the GW potential with any LAMMPS units, but you would need to create your own GW potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.125.6 Related commands

pair_coeff

4.125.7 Default

none

(Gao) Gao and Weber, Nuclear Instruments and Methods in Physics Research B 191 (2012) 504.

4.126 pair_style harmonic/cut command

Accelerator Variants: *harmonic/cut/omp*

4.126.1 Syntax

```
pair_style style
```

- style = *harmonic/cut*

4.126.2 Examples

```
pair_style harmonic/cut
pair_coeff * * 0.2 2.0
pair_coeff 1 1 0.5 2.5
```

4.126.3 Description

Added in version 17Feb2022.

Style *harmonic/cut* computes pairwise repulsive-only harmonic interactions with the formula

$$E = k(r_c - r)^2 \qquad r < r_c$$

where r_c is the cutoff. Note that the usual 1/2 factor is included in k.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- k (energy/distance^2 units)
- r_c (distance units)

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.126.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the k and r_c coefficients can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

Since the potential is zero at and beyond the cutoff parameter by construction, there is no need to support the *pair_modify* shift or tail options for the energy and pressure of the pair interaction.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.126.5 Restrictions

The *harmonic/cut* pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

4.126.6 Related commands

pair_coeff

4.126.7 Default

none

4.127 pair_style hbond/dreiding/lj command

Accelerator Variants: *hbond/dreiding/lj/omp*

4.128 pair_style hbond/dreiding/morse command

Accelerator Variants: *hbond/dreiding/morse/omp*

4.128.1 Syntax

```
pair_style style N inner_distance_cutoff outer_distance_cutoff angle_cutof
```

- style = *hbond/dreiding/lj* or *hbond/dreiding/morse*
- n = cosine angle periodicity
- inner_distance_cutoff = global inner cutoff for Donor-Acceptor interactions (distance units)
- outer_distance_cutoff = global cutoff for Donor-Acceptor interactions (distance units)
- angle_cutoff = global angle cutoff for Acceptor-Hydrogen-Donor
- interactions (degrees)

4.128.2 Examples

```
pair_style hybrid/overlay lj/cut 10.0 hbond/dreiding/lj 4 9.0 11.0 90
pair_coeff 1 2 hbond/dreiding/lj 3 i 9.5 2.75 4 9.0 11.0 90.0

pair_style hybrid/overlay lj/cut 10.0 hbond/dreiding/morse 2 9.0 11.0 90
pair_coeff 1 2 hbond/dreiding/morse 3 i 3.88 1.7241379 2.9 2 9 11 90

labelmap atom 1 C 2 O 3 H
pair_coeff C O hbond/dreiding/morse H i 3.88 1.7241379 2.9 2 9 11 90
```

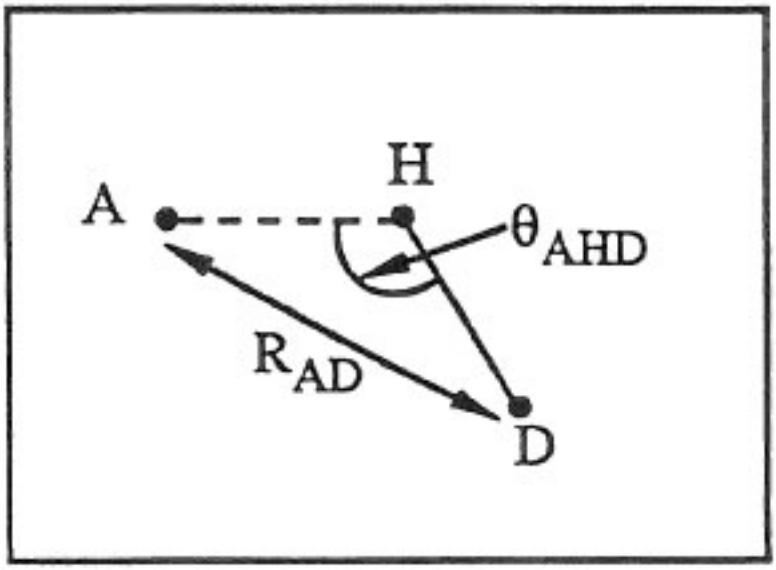
4.128.3 Description

The *hbond/dreiding* styles compute the Acceptor-Hydrogen-Donor (AHD) 3-body hydrogen bond interaction for the *DREIDING* force field, given by:

$$
\begin{aligned}
E =& \left[LJ(r)|Morse(r)\right] && r < r_{\text{in}} \\
=& S(r) * \left[LJ(r)|Morse(r)\right] && r_{\text{in}} < r < r_{\text{out}} \\
=& 0 && r > r_{\text{out}} \\
LJ(r) =& AR^{-12} - BR^{-10} cos^n\theta = \varepsilon \left\{ 5 \left[\frac{\sigma}{r}\right]^{12} - 6 \left[\frac{\sigma}{r}\right]^{10} \right\} cos^n\theta \\
Morse(r) =& D_0 \left\{\chi^2 - 2\chi\right\} cos^n\theta = D_0 \left\{ e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)} \right\} cos^n\theta \\
S(r) =& \frac{\left[r_{\text{out}}^2 - r^2\right]^2 \left[r_{\text{out}}^2 + 2r^2 - 3r_{\text{in}}^2\right]}{\left[r_{\text{out}}^2 - {r_{\text{in}}}^2\right]^3}
\end{aligned}
$$

where r_{in} is the inner spline distance cutoff, r_{out} is the outer distance cutoff, θ_c is the angle cutoff, and n is the cosine periodicity.

Here, r is the radial distance between the donor (D) and acceptor (A) atoms and θ is the bond angle between the acceptor, the hydrogen (H) and the donor atoms:



These 3-body interactions can be defined for pairs of acceptor and donor atoms, based on atom types. For each donor/acceptor atom pair, the third atom in the interaction is a hydrogen permanently bonded to the donor atom, e.g. in a bond list read in from a data file via the *read_data* command. The atom types of possible hydrogen atoms for each donor/acceptor type pair are specified by the *pair_coeff* command (see below).

Style *hbond/dreiding/lj* is the original DREIDING potential of *(Mayo)*. It uses a LJ 12/10 functional for the Donor-Acceptor interactions. To match the results in the original paper, use n = 4.

Style *hbond/dreiding/morse* is an improved version using a Morse potential for the Donor-Acceptor interactions. *(Liu)* showed that the Morse form gives improved results for Dendrimer simulations, when n = 2.

See the *Howto bioFF* page for more information on the DREIDING force field.

Note

Because the Dreiding hydrogen bond potential is only one portion of an overall force field which typically includes other pairwise interactions, it is common to use it as a sub-style in a *pair_style hybrid/overlay* command, where another pair style provides the repulsive core interaction between pairs of atoms, e.g. a 1/r^12 Lennard-Jones repulsion.

Note

When using the hbond/dreiding pair styles with *pair_style hybrid/overlay*, you should explicitly define pair interactions between the donor atom and acceptor atoms, (as well as between these atoms and ALL other atoms in your system). Whenever *pair_style hybrid/overlay* is used, ordinary mixing rules are not applied to atoms like the donor and acceptor atoms because they are typically referenced in multiple pair styles. Neglecting to do this can cause difficult-to-detect physics problems.

Note

In the original Dreiding force field paper 1-4 non-bonded interactions ARE allowed. If this is desired for your model, use the special_bonds command (e.g. "special_bonds lj 0.0 0.0 1.0") to turn these interactions on.

The following coefficients must be defined for pairs of eligible donor/acceptor types via the *pair_coeff* command as in the examples above.

Note

Unlike other pair styles and their associated *pair_coeff* commands, you do not need to specify pair_coeff settings for all possible I,J type pairs. Only I,J type pairs for atoms which act as joint donors/acceptors need to be specified; all other type pairs are assumed to be inactive.

Note

A *pair_coeff* command can be specified multiple times for the same donor/acceptor type pair. This enables multiple hydrogen types to be assigned to the same donor/acceptor type pair. For other pair_styles, if the pair_coeff command is re-used for the same I.J type pair, the settings for that type pair are overwritten. For the hydrogen bond potentials this is not the case; the settings are cumulative. This means the only way to turn off a previous setting, is to re-use the pair_style command and start over.

For the *hbond/dreiding/lj* style the list of coefficients is as follows:

- K = hydrogen atom type = 1 to Ntypes, or type label
- donor flag = *i* or *j*
- ε (energy units)
- σ (distance units)
- n = exponent in formula above
- distance cutoff r_{in} (distance units)
- distance cutoff r_{out} (distance units)
- angle cutoff (degrees)

For the *hbond/dreiding/morse* style the list of coefficients is as follows:

- K = hydrogen atom type = 1 to Ntypes, or type label
- donor flag = *i* or *j*
- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)
- n = exponent in formula above
- distance cutoff r_{in} (distance units)
- distance cutoff r_{out} (distance units)

- angle cutoff (degrees)

A single hydrogen atom type K can be specified, or a wild-card asterisk can be used in place of or in conjunction with the K arguments to select multiple types as hydrogen atoms. This takes the form "*" or "*n" or "n*" or "m*n". See the *pair_coeff* command page for details.

If the donor flag is *i*, then the atom of type I in the pair_coeff command is treated as the donor, and J is the acceptor. If the donor flag is *j*, then the atom of type J in the pair_coeff command is treated as the donor and I is the donor. This option is required because the *pair_coeff* command requires that I <= J.

ε and σ are settings for the hydrogen bond potential based on a Lennard-Jones functional form. Note that sigma is defined as the zero-crossing distance for the potential, not as the energy minimum at $2^{1/6}\sigma$.

D_0 and α and r_0 are settings for the hydrogen bond potential based on a Morse functional form.

The last 3 coefficients for both styles are optional. If not specified, the global n, distance cutoff, and angle cutoff specified in the pair_style command are used. If you wish to only override the second or third optional parameter, you must also specify the preceding optional parameters.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.128.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. You must explicitly identify each donor/acceptor type pair.

These styles do not support the *pair_modify* shift option for the energy of the interactions.

The *pair_modify* table option is not relevant for these pair styles.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles do not write their information to *binary restart files*, so pair_style and pair_coeff commands need to be re-specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

These pair styles tally a count of how many hydrogen bonding interactions they calculate each timestep and the hbond energy. These quantities can be accessed via the *compute pair* command as a vector of values of length 2.

To print these quantities to the log file (with a descriptive column heading) the following commands could be included in an input script:

```
compute hb all pair hbond/dreiding/lj
variable n_hbond equal c_hb[1] #number hbonds
variable E_hbond equal c_hb[2] #hbond energy
thermo_style custom step temp epair v_E_hbond
```

4.128.5 Restrictions

This pair style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

4.128.6 Related commands

pair_coeff

4.128.7 Default

none

(Mayo) Mayo, Olfason, Goddard III, J Phys Chem, 94, 8897-8909 (1990).

(Liu) Liu, Bryantsev, Diallo, Goddard III, J. Am. Chem. Soc 131 (8) 2798 (2009)

4.129 pair_style hdnnp command

4.129.1 Syntax

```
pair_style hdnnp cutoff keyword value ...
```

- cutoff = short-range cutoff of HDNNP (maximum symmetry function cutoff radius)
- zero or more keyword/value pairs may be appended
- keyword = *dir* or *showew* or *showewsum* or *maxew* or *resetew* or *cflength* or *cfenergy*
- value depends on the preceding keyword:

```
dir value = directory
   directory = Path to HDNNP configuration files
showew value = yes or no
showewsum value = summary
        summary = Write EW summary every this many timesteps (0 turns summary off)
maxew value = threshold
     threshold = Maximum number of EWs allowed
resetew value = yes or no
cflength value = length
        length = Length unit conversion factor
cfenergy value = energy
        energy = Energy unit conversion factor
```

4.129.2 Examples

```
pair_style hdnnp 6.35 showew yes showewsum 100 maxew 1000 resetew yes cflength 1.8897261328␣
↪cfenergy 0.0367493254
pair_coeff * * H O

pair_style hdnnp 6.01 dir "./" showewsum 10000
pair_coeff * * S Cu NULL Cu
```

4.129.3 Description

This pair style adds an interaction based on the high-dimensional neural network potential (HDNNP) method as presented in *(Behler and Parrinello 2007)*. HDNNPs are machine learning potentials which require careful training of neural networks prior to application in MD simulations. The pair style uses an interface to the *n2p2* library *(Singraber, Behler and Dellago 2019)* which is available on Github here. Please see the *n2p2* documentation for further details. *n2p2* (and hence this pair style) is compatible with neural network potentials trained with its own tools *(Singraber et al 2019)* and with RuNNer.

Only a single *pair_coeff* command with two asterisk wild-cards is used with this pair style. Its additional arguments define the mapping of LAMMPS atom types to n2p2 elements.

```
pair_coeff * * H O
```

In the above example LAMMPS types 1 and 2 are mapped to the elements "H" and "O" in n2p2, respectively. Multiple types may map to the same element, or some types may not be mapped at all. For example, if the LAMMPS simulation has four atom types, the command

```
pair_coeff * * H H O NULL
```

maps atom types 1 and 2 to the element "H", type 3 to "O" and type 4 is not mapped (indicated by NULL). Atoms mapped to NULL are ignored by the HDNNP calculation, i.e. they do not contribute in any way to the evaluation of HDNNP energies and forces. This may be useful in a setup with *hybrid pair styles*.

The mandatory pair style argument *cutoff* must match the short-range cutoff radius of the HDNNP. This corresponds to the maximum cutoff radius of all symmetry functions (the atomic environment descriptors of HDNNPs) used.

Note

The cutoff must be given in LAMMPS length units, even if the neural network potential has been trained using a different unit system (see remarks about the *cflength* and *cfenergy* keywords below for details).

The numeric value may be slightly larger than the actual maximum symmetry function cutoff radius (to account for rounding errors when converting units), but must not be smaller.

Use the *dir* keyword to specify the directory containing the HDNNP configuration files. The directory must contain input.nn with neural network and symmetry function setup, scaling.data with symmetry function scaling data and weights.???.data with weight parameters for each element.

The keyword *showew* can be used to turn on/off the display of extrapolation warnings (EWs) which are issued whenever a symmetry function value is out of bounds defined by minimum/maximum values in scaling.data. An extrapolation warning may look like this:

```
### NNP EXTRAPOLATION WARNING ### STRUCTURE:      0 ATOM:      119 ELEMENT:␣
↪Cu SYMFUNC:   32 TYPE:  3 VALUE:  2.166E-02 MIN:  2.003E-05 MAX:  1.756E-02
```

stating that the value 2.166E-02 of symmetry function 32 of type 3 (Narrow Angular symmetry function), element Cu (see the log file for a symmetry function listing) was out of bounds (maximum in scaling.data is 1.756E-02) for atom 119. Here, the atom index refers to the LAMMPS tag (global index) and the structure index is used to print out the MPI rank the atom belongs to.

Note

The *showew* keyword should only be set to *yes* for debugging purposes. Extrapolation warnings may add lots of overhead as they are communicated each timestep. Also, if the simulation is run in a region where the HDNNP was not correctly trained, lots of extrapolation warnings may clog log files and the console. In a production run use *showewsum* instead.

The keyword *showewsum* can be used to get an overview of extrapolation warnings occurring during an MD simulation. The argument specifies the interval at which extrapolation warning summaries are displayed and logged. An EW summary may look like this:

```
### NNP EW SUMMARY ### TS:        100 EW         11 EWPERSTEP  1.100E-01
```

Here, at timestep 100 the occurrence of 11 extrapolation warnings since the last summary is reported, which corresponds to an EW rate of 0.11 per timestep. Setting *showewsum* to 0 deactivates the EW summaries.

A maximum number of allowed extrapolation warnings can be specified with the *maxew* keyword. If the number of EWs exceeds the *maxew* argument the simulation is stopped. Note however that this is merely an approximate threshold since the check is only performed at the end of each timestep and each MPI process counts individually to minimize communication overhead.

The keyword *resetew* alters the behavior of the above mentioned *maxew* threshold. If *resetew* is set to *yes* the threshold is applied on a per-timestep basis and the internal EW counters are reset at the beginning of each timestep. With *resetew* set to *no* the counters accumulate EWs along the whole trajectory.

If the training of a neural network potential has been performed with different physical units for length and energy than those set in LAMMPS, it is still possible to use the potential when the unit conversion factors are provided via the *cflength* and *cfenergy* keywords. If for example, the HDNNP was parameterized with Bohr and Hartree training data and symmetry function parameters (i.e. distances and energies in "input.nn" are given in Bohr and Hartree) but LAMMPS is set to use *metal* units (Angstrom and eV) the correct conversion factors are:

```
cflength 1.8897261328

cfenergy 0.0367493254
```

Thus, arguments of *cflength* and *cfenergy* are the multiplicative factors required to convert lengths and energies given in LAMMPS units to respective quantities in native HDNNP units (1 Angstrom = 1.8897261328 Bohr, 1 eV = 0.0367493254 Hartree).

4.129.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. The *pair_coeff* command should only be invoked with asterisk wild cards (see above).

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.129.5 Restrictions

This pair style is part of the ML-HDNNP package. It is only enabled if LAMMPS was built with that package. See the *Build package* doc page for more info.

Please report bugs and feature requests to the n2p2 GitHub issue page.

Related commands

pair_coeff, *pair_hybrid*, *units*

Default

The default options are *dir* = "hdnnp/", *showew* = yes, *showewsum* = 0, *maxew* = 0, *resetew* = no, *cflength* = 1.0, *cfenergy* = 1.0.

(Behler and Parrinello 2007) Behler, J.; Parrinello, M. Phys. Rev. Lett. 2007, 98 (14), 146401.

(Singraber, Behler and Dellago 2019) Singraber, A.; Behler, J.; Dellago, C. J., Chem. Theory Comput. 2019, 15 (3), 1827-1840

(Singraber et al 2019) Singraber, A.; Morawietz, T.; Behler, J.; Dellago, C., J. Chem. Theory Comput. 2019, 15 (5), 3075-3092.

4.130 pair_style hybrid command

Accelerator Variants: *hybrid/kk*, *hybrid/omp*

4.131 pair_style hybrid/molecular command

Accelerator Variant: *hybrid/molecular/omp*

4.132 pair_style hybrid/overlay command

Accelerator Variants: *hybrid/overlay/kk*, *hybrid/overlay/omp*

4.133 pair_style hybrid/scaled command

Accelerator Variant: *hybrid/scaled/omp*

4.133.1 Syntax

```
pair_style hybrid style1 args style2 args ...
pair_style hybrid/molecular factor1 style1 args factor2 style 2 args
pair_style hybrid/overlay style1 args style2 args ...
pair_style hybrid/scaled factor1 style1 args factor2 style 2 args ...
```

- style1,style2 = list of one or more pair styles and their arguments
- factor1,factor2 = scale factors for pair styles, may be a variable

4.133.2 Examples

```
pair_style hybrid lj/cut/coul/cut 10.0 eam lj/cut 5.0
pair_coeff 1*2 1*2 eam niu3
pair_coeff 3 3 lj/cut/coul/cut 1.0 1.0
pair_coeff 1*2 3 lj/cut 0.5 1.2

pair_style hybrid/overlay lj/cut 2.5 coul/long 2.0
pair_coeff * * lj/cut 1.0 1.0
pair_coeff * * coul/long

pair_style hybrid/scaled 0.5 tersoff 0.5 sw
pair_coeff * * tersoff Si.tersoff Si
pair_coeff * * sw Si.sw Si

pair_style hybrid/molecular lj/cut 2.5 lj/cut 2.5
pair_coeff * * lj/cut 1 1.0 1.0
pair_coeff * * lj/cut 2 1.5 1.0

variable one equal ramp(1.0,0.0)
variable two equal 1.0-v_one
pair_style hybrid/scaled v_one lj/cut 2.5 v_two morse 2.5
pair_coeff 1 1 lj/cut 1.0 1.0 2.5
pair_coeff 1 1 morse 1.0 1.0 1.0 2.5
```

4.133.3 Description

The *hybrid*, *hybrid/overlay*, *hybrid/molecular*, and *hybrid/scaled* styles enable the use of multiple pair styles in one simulation. With the *hybrid* style, exactly one pair style is assigned to each pair of atom types. With the *hybrid/overlay* and *hybrid/scaled* styles, one or more pair styles can be assigned to each pair of atom types. With the hybrid/molecular style, pair styles are assigned to either intra- or inter-molecular interactions.

The assignment of pair styles to type pairs is made via the *pair_coeff* command. The major difference between the *hybrid/overlay* and *hybrid/scaled* styles is that the *hybrid/scaled* adds a scale factor for each sub-style contribution to forces, energies and stresses. Because of the added complexity, the *hybrid/scaled* style has more overhead and thus may be slower than *hybrid/overlay*.

The *hybrid/molecular* pair style accepts *only* two sub-styles: the first is assigned to intra-molecular interactions (i.e. both atoms have the same molecule ID), the second to inter-molecular interactions (i.e. interacting atoms have different molecule IDs).

Here are two examples of hybrid simulations. The *hybrid* style could be used for a simulation of a metal droplet on a LJ surface. The metal atoms interact with each other via an *eam* potential, the surface atoms interact with each other via a *lj/cut* potential, and the metal/surface interaction is also computed via a *lj/cut* potential. The *hybrid/overlay* style could be used as in the second example above, where multiple potentials are superimposed in an additive fashion to compute the interaction between atoms. In this example, using *lj/cut* and *coul/long* together gives the same result as if the *lj/cut/coul/long* potential were used by itself. In this case, it would be more efficient to use the single combined potential, but in general any combination of pair potentials can be used together in to produce an interaction that is not encoded in any single pair_style file, e.g. adding Coulombic forces between granular particles. Another limitation of using the *hybrid/overlay* variant, that it does not generate *lj/cut* parameters for mixed atom types from a mixing rule due to restrictions discussed below.

If the *hybrid/scaled* style is used instead of *hybrid/overlay*, contributions from sub-styles are weighted by their scale factors, which may be fractional or even negative. Furthermore the scale factors may be variables that may change during a simulation. This enables switching smoothly between two different pair styles or two different parameter sets during a run in a similar fashion as could be done with *fix adapt* or *fix alchemy*.

All pair styles that will be used are listed as "sub-styles" following the *hybrid* or *hybrid/overlay* keyword, in any order. In case of the *hybrid/scaled* pair style, each sub-style is prefixed with a scale factor. The scale factor is either a floating point number or an equal style (or equivalent) variable. Each sub-style's name is followed by its usual arguments, as illustrated in the examples above. See the doc pages of the individual pair styles for a listing and explanation of the appropriate arguments for them.

Note that an individual pair style can be used multiple times as a sub-style. For efficiency reasons this should only be done if your model requires it. E.g. if you have different regions of Si and C atoms and wish to use a Tersoff potential for pure Si for one set of atoms, and a Tersoff potential for pure C for the other set (presumably with some third potential for Si-C interactions), then the sub-style *tersoff* could be listed twice. But if you just want to use a Lennard-Jones or other pairwise potential for several different atom type pairs in your model, then you should just list the sub-style once and use the pair_coeff command to assign parameters for the different type pairs.

Note

There is one exception to this option to list an individual pair style multiple times: GPU-enabled pair styles in the GPU package. This is because the GPU package currently assumes that only one instance of a pair style is being used.

In the pair_coeff commands, the name of a pair style must be added after the I,J type specification, with the remaining coefficients being those appropriate to that style. If the pair style is used multiple times in the pair_style command, then an additional numeric argument must also be specified which is a number from 1 to M where M is the number of times the sub-style was listed in the pair style command. The extra number indicates which instance of the sub-style these coefficients apply to.

For example, consider a simulation with 3 atom types: types 1 and 2 are Ni atoms, type 3 are LJ atoms with charges. The following commands would set up a hybrid simulation:

```
pair_style hybrid eam/alloy lj/cut/coul/cut 10.0 lj/cut 8.0
pair_coeff * * eam/alloy nialhjea Ni Ni NULL
pair_coeff 3 3 lj/cut/coul/cut 1.0 1.0
pair_coeff 1*2 3 lj/cut 0.8 1.3
```

As an example of using the same pair style multiple times, consider a simulation with 2 atom types. Type 1 is Si, type 2 is C. The following commands would model the Si atoms with Tersoff, the C atoms with Tersoff, and the cross-interactions with Lennard-Jones:

```
pair_style hybrid lj/cut 2.5 tersoff tersoff
pair_coeff * * tersoff 1 Si.tersoff Si NULL
pair_coeff * * tersoff 2 C.tersoff NULL C
pair_coeff 1 2 lj/cut 1.0 1.5
```

It is not recommended to read pair coefficients for a hybrid style from a "Pair Coeffs" or "PairIJ Coeffs" section of a data file via the *read_data* command, since those sections expect a fixed number of lines, either one line per atom type or one line pair pair of atom types, respectively. When reading from a data file, the lines of the "Pair Coeffs" and "PairIJ Coeffs" are changed in the same way as the *pair_coeff* command, i.e. the name of the pair style to which the parameters apply must follow the atom type (or atom types), e.g.

```
Pair Coeffs

1 lj/cut/coul/cut 1.0 1.0
...

PairIJ Coeffs

1 1 lj/cut/coul/cut 1.0 1.0
...
```

Note that the pair_coeff command for some potentials such as *pair_style eam/alloy* includes a mapping specification of elements to all atom types, which in the hybrid case, can include atom types not assigned to the *eam/alloy* potential. The NULL keyword is used by many such potentials (eam/alloy, Tersoff, AIREBO, etc), to denote an atom type that will be assigned to a different sub-style.

For the *hybrid* style, each atom type pair I,J is assigned to exactly one sub-style. Just as with a simulation using a single pair style, if you specify the same atom type pair in a second pair_coeff command, the previous assignment will be overwritten.

For the *hybrid/overlay* and *hybrid/scaled* styles, each atom type pair I,J can be assigned to one or more sub-styles. If you specify the same atom type pair in a second pair_coeff command with a new sub-style, then the second sub-style is added to the list of potentials that will be calculated for two interacting atoms of those types. If you specify the same atom type pair in a second pair_coeff command with a sub-style that has already been defined for that pair of atoms, then the new pair coefficients simply override the previous ones, as in the normal usage of the pair_coeff command. E.g. these two sets of commands are the same:

```
pair_style lj/cut 2.5
pair_coeff * * 1.0 1.0
pair_coeff 2 2 1.5 0.8

pair_style hybrid/overlay lj/cut 2.5
```

(continues on next page)

(continued from previous page)

```
pair_coeff * * lj/cut 1.0 1.0
pair_coeff 2 2 lj/cut 1.5 0.8
```

Coefficients must be defined for each pair of atoms types via the *pair_coeff* command as described above, or in the "Pair Coeffs" or "PairIJ Coeffs" section of the data file read by the *read_data* command, or by mixing as described below.

For all of the *hybrid*, *hybrid/overlay*, and *hybrid/scaled* styles, every atom type pair I,J (where I <= J) must be assigned to at least one sub-style via the *pair_coeff* command as in the examples above, or in the data file read by the *read_data*, or by mixing as described below. Also all sub-styles must be used at least once in a *pair_coeff* command.

Warning

With hybrid pair styles the use of mixing to generate pair coefficients is significantly limited compared to the individual pair styles. LAMMPS **never** performs mixing of parameters from different sub-styles, **even** if they use the same type of coefficients, e.g. contain a Lennard-Jones potential variant. Those parameters must be provided explicitly. Also for *hybrid/overlay* and *hybrid/scaled* mixing is **only** performed for pairs of atom types for which only a single pair style is assigned.

Thus it is strongly recommended to provide all mixed terms explicitly. For non-hybrid styles those could be generated and written out using the *write_coeff command* and then edited as needed to comply with the requirements for hybrid styles as explained above.

If you want there to be no interactions between a particular pair of atom types, you have 3 choices. You can assign the pair of atom types to some sub-style and use the *neigh_modify exclude type* command. You can assign it to some sub-style and set the coefficients so that there is effectively no interaction (e.g. epsilon = 0.0 in a LJ potential). Or, for *hybrid*, *hybrid/overlay*, or *hybrid/scaled* simulations, you can use this form of the pair_coeff command in your input script or the "PairIJ Coeffs" section of your data file:

```
pair_coeff 2 3 none
```

or this form in the "Pair Coeffs" section of the data file:

```
3 none
```

If an assignment to *none* is made in a simulation with the *hybrid/overlay* or *hybrid/scaled* pair style, it wipes out all previous assignments of that pair of atom types to sub-styles.

Note that you may need to use an *atom_style* hybrid command in your input script, if atoms in the simulation will need attributes from several atom styles, due to using multiple pair styles with different requirements.

Different force fields (e.g. CHARMM vs. AMBER) may have different rules for applying exclusions or weights that change the strength of pairwise non-bonded interactions between pairs of atoms that are also 1-2, 1-3, and 1-4 neighbors in the molecular bond topology. This is normally a global setting defined the *special_bonds* command. However, different weights can be assigned to different hybrid sub-styles via the *pair_modify special* command. This allows multiple force fields to be used in a model of a hybrid system, however, there is no consistent approach to determine parameters automatically for the interactions **between** atoms of the two force fields, thus this approach this is only recommended when particles described by the different force fields do not mix.

Here is an example for combining CHARMM and AMBER: The global *amber* setting sets the 1-4 interactions to non-zero scaling factors and then overrides them with 0.0 only for CHARMM:

```
special_bonds amber
pair_style hybrid lj/charmm/coul/long 8.0 10.0 lj/cut/coul/long 10.0
pair_modify pair lj/charmm/coul/long special lj/coul 0.0 0.0 0.0
```

This input achieves the same effect:

```
special_bonds 0.0 0.0 0.1
pair_style hybrid lj/charmm/coul/long 8.0 10.0 lj/cut/coul/long 10.0
pair_modify pair lj/cut/coul/long special lj 0.0 0.0 0.5
pair_modify pair lj/cut/coul/long special coul 0.0 0.0 0.83333333
pair_modify pair lj/charmm/coul/long special lj/coul 0.0 0.0 0.0
```

Here is an example for combining Tersoff with OPLS/AA based on a data file that defines bonds for all atoms where - for the Tersoff part of the system - the force constants for the bonded interactions have been set to 0. Note the global settings are effectively *lj/coul 0.0 0.0 0.5* as required for OPLS/AA:

```
special_bonds lj/coul 1e-20 1e-20 0.5
pair_style hybrid tersoff lj/cut/coul/long 12.0
pair_modify pair tersoff special lj/coul 1.0 1.0 1.0
```

For use with the various *compute */tally* computes, the *pair_modify compute/tally* command can be used to selectively turn off processing of the compute tally styles, for example, if those pair styles (e.g. many-body styles) do not support this feature.

See the *pair_modify* page for details on the specific syntax, requirements and restrictions.

The potential energy contribution to the overall system due to an individual sub-style can be accessed and output via the *compute pair* command. Note that in the case of pair style *hybrid/scaled* this is the **unscaled** potential energy of the selected sub-style.

Note

Several of the potentials defined via the pair_style command in LAMMPS are really many-body potentials, such as Tersoff, AIREBO, MEAM, ReaxFF, etc. The way to think about using these potentials in a hybrid setting is as follows.

A subset of atom types is assigned to the many-body potential with a single *pair_coeff* command, using "* *" to include all types and the NULL keywords described above to exclude specific types not assigned to that potential. If types 1,3,4 were assigned in that way (but not type 2), this means that all many-body interactions between all atoms of types 1,3,4 will be computed by that potential. Pair_style hybrid allows interactions between type pairs 2-2, 1-2, 2-3, 2-4 to be specified for computation by other pair styles. You could even add a second interaction for 1-1 to be computed by another pair style, assuming pair_style hybrid/overlay is used.

But you should not, as a general rule, attempt to exclude the many-body interactions for some subset of the type pairs within the set of 1,3,4 interactions, e.g. exclude 1-1 or 1-3 interactions. That is not conceptually well-defined for many-body interactions, since the potential will typically calculate energies and foces for small groups of atoms, e.g. 3 or 4 atoms, using the neighbor lists of the atoms to find the additional atoms in the group.

However, you can still use the pair_coeff none setting or the *neigh_modify exclude* command to exclude certain type pairs from the neighbor list that will be passed to a many-body sub-style. This will alter the calculations made by a many-body potential beyond the specific pairs, since it builds its list of 3-body, 4-body, etc interactions from the pair

lists. You will need to think **carefully** as to whether excluding such pairs produces a physically meaningful result for your model.

For example, imagine you have two atom types in your model, type 1 for atoms in one surface, and type 2 for atoms in the other, and you wish to use a Tersoff potential to compute interactions within each surface, but not between the surfaces. Then either of these two command sequences would implement that model:

```
pair_style hybrid tersoff
pair_coeff * * tersoff SiC.tersoff C C
pair_coeff 1 2 none

pair_style tersoff
pair_coeff * * SiC.tersoff C C
neigh_modify exclude type 1 2
```

Either way, only neighbor lists with 1-1 or 2-2 interactions would be passed to the Tersoff potential, which means it would compute no 3-body interactions containing both type 1 and 2 atoms.

Here is another example to use 2 many-body potentials together in an overlapping manner using hybrid/overlay. Imagine you have CNT (C atoms) on a Si surface. You want to use Tersoff for Si/Si and Si/C interactions, and AIREBO for C/C interactions. Si atoms are type 1; C atoms are type 2. Something like this will work:

```
pair_style hybrid/overlay tersoff airebo 3.0
pair_coeff * * tersoff SiC.tersoff.custom Si C
pair_coeff * * airebo CH.airebo NULL C
```

Note that to prevent the Tersoff potential from computing C/C interactions, you would need to **modify** the SiC.tersoff potential file to turn off C/C interaction, i.e. by setting the appropriate coefficients to 0.0.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

Note

Since the *hybrid*, *hybrid/overlay*, *hybrid/scaled* styles delegate computation to the individual sub-styles, the suffix versions of the *hybrid* and *hybrid/overlay* styles are used to propagate the corresponding suffix to all sub-styles, if those versions exist. Otherwise the non-accelerated version will be used. The individual accelerated sub-styles are part of the GPU, KOKKOS, INTEL, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

4.133.4 Mixing, shift, table, tail correction, restart, rRESPA info

Any pair potential settings made via the *pair_modify* command are passed along to all sub-styles of the hybrid potential.

For atom type pairs I,J and I != J, if the sub-style assigned to I,I and J,J is the same, and if the sub-style allows for mixing, then the coefficients for I,J can be mixed. This means you do not have to specify a pair_coeff command for I,J since the I,J type pair will be assigned automatically to the sub-style defined for both I,I and J,J and its coefficients generated by the mixing rule used by that sub-style. For the *hybrid/overlay* and *hybrid/scaled* style, there is an additional requirement that both the I,I and J,J pairs are assigned to a single sub-style. If this requirement is not met, no I,J coeffs will be generated, even if the sub-styles support mixing, and I,J pair coefficients must be explicitly defined.

See the *pair_modify* command for details of mixing rules. See the See the page for the sub-style to see if allows for mixing.

The hybrid pair styles supports the *pair_modify* shift, table, and tail options for an I,J pair interaction, if the associated sub-style supports it.

For the hybrid pair styles, the list of sub-styles and their respective settings are written to *binary restart files*, so a *pair_style* command does not need to specified in an input script that reads a restart file. However, the coefficient information is not stored in the restart file. The same is true for *data files*. Thus, pair_coeff commands need to be re-specified in the restart input script. For pair style *hybrid/scaled* also the names of any variables used as scale factors are restored, but not the variables themselves, so those may need to be redefined when continuing from a restart.

These pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, if their sub-styles do.

4.133.5 Restrictions

When using a long-range Coulombic solver (via the *kspace_style* command) with a hybrid pair_style, one or more sub-styles will be of the "long" variety, e.g. *lj/cut/coul/long* or *buck/coul/long*. You must ensure that the short-range Coulombic cutoff used by each of these long pair styles is the same or else LAMMPS will generate an error.

Pair style *hybrid/scaled* currently only works for non-accelerated pair styles and pair styles from the OPT package.

Pair style *hybrid/molecular* is not compatible with manybody potentials.

When using pair styles from the GPU package they must not be listed multiple times. LAMMPS will detect this and abort.

4.133.6 Related commands

pair_coeff

4.133.7 Default

none

4.134 pair_style ilp/graphene/hbn command

Accelerator Variant: *ilp/graphene/hbn/opt*

4.134.1 Syntax

```
pair_style [hybrid/overlay ...] ilp/graphene/hbn cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.134.2 Examples

```
pair_style  hybrid/overlay ilp/graphene/hbn 16.0 1
pair_coeff  * * ilp/graphene/hbn  BNCH.ILP B N C

pair_style  hybrid/overlay rebo tersoff ilp/graphene/hbn 16.0 coul/shield 16.0
pair_coeff  * * rebo              CH.rebo     NULL NULL C
pair_coeff  * * tersoff           BNC.tersoff B    N    NULL
pair_coeff  * * ilp/graphene/hbn  BNCH.ILP    B    N    C
pair_coeff  1 1 coul/shield 0.70
pair_coeff  1 2 coul/shield 0.695
pair_coeff  2 2 coul/shield 0.69
```

4.134.3 Description

The *ilp/graphene/hbn* style computes the registry-dependent interlayer potential (ILP) potential as described in *(Leven1)*, *(Leven2)* and *(Maaravi)*. The normals are calculated in the way as described in *(Kolmogorov)*.

$$
\begin{aligned}
E =& \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& \mathrm{Tap}(r_{ij}) \left\{ e^{-\alpha(r_{ij}/\beta - 1)} \left[\varepsilon + f(\rho_{ij}) + f(\rho_{ji})\right] - \frac{1}{1 + e^{-d[(r_{ij}/(s_R \cdot r^{eff})) - 1]}} \cdot \frac{C_6}{r_{ij}^6} \right\} \\
\rho_{ij}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_i)^2 \\
\rho_{ji}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_j)^2 \\
f(\rho) =& C e^{-(\rho/\delta)^2} \\
\mathrm{Tap}(r_{ij}) =& 20 \left(\frac{r_{ij}}{R_{cut}}\right)^7 - 70 \left(\frac{r_{ij}}{R_{cut}}\right)^6 + 84 \left(\frac{r_{ij}}{R_{cut}}\right)^5 - 35 \left(\frac{r_{ij}}{R_{cut}}\right)^4 + 1
\end{aligned}
$$

Where $\mathrm{Tap}(r_{ij})$ is the taper function which provides a continuous cutoff (up to third derivative) for interatomic separations larger than r_c *(Maaravi)*. The definitions of each parameter in the above equation can be found in *(Leven1)* and *(Maaravi)*.

It is important to include all the pairs to build the neighbor list for calculating the normals.

Note

This potential (ILP) is intended for interlayer interactions between two different layers of graphene, hexagonal boron nitride (h-BN) and their hetero-junction. To perform a realistic simulation, this potential must be used in combination with intralayer potential, such as *AIREBO* or *Tersoff* potential. To keep the intralayer properties unaffected, the interlayer interaction within the same layers should be avoided. Hence, each atom has to have a layer identifier such that atoms residing on the same layer interact via the appropriate intralayer potential and atoms residing on different layers interact via the ILP. Here, the molecule id is chosen as the layer identifier, thus a data file with the "full" atom style is required to use this potential.

The parameter file (e.g. BNCH.ILP), is intended for use with *metal* *units*, with energies in meV. Two additional parameters, *S*, and *rcut* are included in the parameter file. *S* is designed to facilitate scaling of energies. *rcut* is designed to build the neighbor list for calculating the normals for each atom pair.

Note

The parameters presented in the parameter file (e.g. BNCH.ILP), are fitted with taper function by setting the cutoff equal to 16.0 Angstrom. Using different cutoff or taper function should be careful. The parameters for atoms pairs between Boron and Nitrogen are fitted with a screened Coulomb interaction *coul/shield*. Therefore, to simulated the properties of h-BN correctly, this potential must be used in combination with the pair style *coul/shield*.

Note

Four new sets of parameters of ILP for 2D layered Materials with bilayer and bulk configurations are presented in *(Ouyang1)* and *(Ouyang2)*, respectively. These parameters provide a good description in both short- and long-range interaction regimes. While the old ILP parameters published in *(Leven2)* and *(Maaravi)* are only suitable for long-range interaction regime. This feature is essential for simulations in high pressure regime (i.e., the interlayer distance is smaller than the equilibrium distance). The benchmark tests and comparison of these parameters can be found in *(Ouyang1)* and *(Ouyang2)*.

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using pair_style *none*.

This pair style tallies a breakdown of the total interlayer potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 2. The 2 values correspond to the following sub-categories:

1. *E_vdW* = vdW (attractive) energy
2. *E_Rep* = Repulsive energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair ilp/graphene/hbn
variable Evdw  equal c_0[1]
variable Erep  equal c_0[2]
thermo_style custom step temp epair v_Erep v_Evdw
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.134.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.134.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the newton setting to be *on* for pair interactions.

The BNCH.ILP potential file provided with LAMMPS (see the potentials directory) are parameterized for *metal* units. You can use this potential with any LAMMPS units, but you would need to create your own custom BNCH.ILP potential file with coefficients listed in the appropriate units, if your simulation does not use *metal* units.

4.134.6 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style ilp_tmd*, *pair_style saip_metal*, *pair_style pair_kolmogorov_crespi_z*, *pair_style pair_kolmogorov_crespi_full*, *pair_style pair_lebedeva_z*, *pair_style pair_coul_shield*.

4.134.7 Default

tap_flag = 1

(Ouyang1) W. Ouyang, D. Mandelli, M. Urbakh and O. Hod, Nano Lett. 18, 6009-6016 (2018).

(Ouyang2) W. Ouyang et al., J. Chem. Theory Comput. 16(1), 666-676 (2020).

(Leven1) I. Leven, I. Azuri, L. Kronik and O. Hod, J. Chem. Phys. 140, 104106 (2014).

(Leven2) I. Leven et al, J. Chem.Theory Comput. 12, 2896-905 (2016).

(Maaravi) T. Maaravi et al, J. Phys. Chem. C 121, 22826-22835 (2017).

(Kolmogorov) A. N. Kolmogorov, V. H. Crespi, Phys. Rev. B 71, 235415 (2005).

4.135 pair_style ilp/tmd command

Accelerator Variant: *ilp/tmd/opt*

4.135.1 Syntax

```
pair_style [hybrid/overlay ...] ilp/tmd cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.135.2 Examples

```
pair_style  hybrid/overlay ilp/tmd 16.0 1
pair_coeff  * * ilp/tmd  TMD.ILP Mo S S

pair_style  hybrid/overlay sw/mod sw/mod ilp/tmd 16.0
pair_coeff  * * sw/mod 1  tmd.sw.mod Mo S S NULL NULL NULL
pair_coeff  * * sw/mod 2  tmd.sw.mod NULL NULL NULL W Se Se
pair_coeff  * * ilp/tmd   TMD.ILP   Mo S S W Se Se
```

4.135.3 Description

Added in version 17Feb2022.

The *ilp/tmd* style computes the registry-dependent interlayer potential (ILP) potential for transition metal dichalcogenides (TMD) as described in *(Ouyang7)* and *(Jiang)*.

$$
\begin{aligned}
E =& \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& \mathrm{Tap}(r_{ij}) \left\{ e^{-\alpha(r_{ij}/\beta - 1)} \left[\varepsilon + f(\rho_{ij}) + f(\rho_{ji})\right] - \frac{1}{1 + e^{-d\left[(r_{ij}/(s_R \cdot r^{eff})) - 1\right]}} \cdot \frac{C_6}{r_{ij}^6} \right\} \\
\rho_{ij}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_i)^2 \\
\rho_{ji}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_j)^2 \\
f(\rho) =& C e^{-(\rho/\delta)^2} \\
\mathrm{Tap}(r_{ij}) =& 20 \left(\frac{r_{ij}}{R_{cut}}\right)^7 - 70 \left(\frac{r_{ij}}{R_{cut}}\right)^6 + 84 \left(\frac{r_{ij}}{R_{cut}}\right)^5 - 35 \left(\frac{r_{ij}}{R_{cut}}\right)^4 + 1
\end{aligned}
$$

Where $\mathrm{Tap}(r_{ij})$ is the taper function which provides a continuous cutoff (up to third derivative) for interatomic separations larger than r_c *pair_style ilp_graphene_hbn*.

It is important to include all the pairs to build the neighbor list for calculating the normals.

Note

Since each MX2 (M = Mo, W and X = S, Se Te) layer contains two sub-layers of X atoms and one sub-layer of M atoms, the definition of the normal vectors used for graphene and h-BN is no longer valid for TMDs. In *(Ouyang7)*, a

new definition is proposed, where for each atom *i*, its six nearest neighboring atoms belonging to the same sub-layer are chosen to define the normal vector *{bf n}_i*.

The parameter file (e.g. TMD.ILP), is intended for use with *metal* *units*, with energies in meV. Two additional parameters, *S*, and *rcut* are included in the parameter file. *S* is designed to facilitate scaling of energies. *rcut* is designed to build the neighbor list for calculating the normals for each atom pair.

Note

The parameters presented in the parameter file (e.g. TMD.ILP), are fitted with taper function by setting the cutoff equal to 16.0 Angstrom. Using different cutoff or taper function should be careful. These parameters provide a good description in both short- and long-range interaction regimes. This feature is essential for simulations in high pressure regime (i.e., the interlayer distance is smaller than the equilibrium distance). The benchmark tests and comparison of these parameters can be found in *(Ouyang7)*.

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using pair_style *none*.

This pair style tallies a breakdown of the total interlayer potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 2. The 2 values correspond to the following sub-categories:

1. *E_vdW* = vdW (attractive) energy
2. *E_Rep* = Repulsive energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair ilp/tmd
variable Evdw  equal c_0[1]
variable Erep  equal c_0[2]
thermo_style custom step temp epair v_Erep v_Evdw
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.135.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.135.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the newton setting to be *on* for pair interactions.

The TMD.ILP potential file provided with LAMMPS (see the potentials directory) are parameterized for *metal* units. You can use this potential with any LAMMPS units, but you would need to create your own custom TMD.ILP potential file with coefficients listed in the appropriate units, if your simulation does not use *metal* units.

4.135.6 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style saip_metal*, *pair_style ilp_graphene_hbn*, *pair_style pair_kolmogorov_crespi_z*, *pair_style pair_kolmogorov_crespi_full*, *pair_style pair_lebedeva_z*, *pair_style pair_coul_shield*.

4.135.7 Default

tap_flag = 1

(Ouyang7) W. Ouyang, et al., J. Chem. Theory Comput. 17, 7237 (2021).

(Jiang) W. Jiang, et al., J. Phys. Chem. A, 127, 46, 9820-9830 (2023).

4.136 pair_style kim command

4.136.1 Syntax

```
pair_style kim model
```

model = name of a KIM model (the KIM ID for models archived in OpenKIM)

4.136.2 Examples

```
pair_style kim SW_StillingerWeber_1985_Si__MO_405512056662_005
pair_coeff * * Si
```

4.136.3 Description

This pair style is a wrapper on the Open Knowledgebase of Interatomic Models (OpenKIM) repository of interatomic potentials to enable their use in LAMMPS scripts.

The preferred interface for using interatomic models archived in OpenKIM is the *kim command* interface. That interface supports both "KIM Portable Models" (PMs) that conform to the KIM API Portable Model Interface (PMI) and can be used by any simulation code that conforms to the KIM API/PMI, and "KIM Simulator Models" (SMs) that are natively implemented within a single simulation code (like LAMMPS) and can only be used with it. The *pair_style kim* command is limited to KIM PMs. It is used by the *kim command* interface as needed.

Note

Since *pair_style kim* is called by *kim interactions* as needed, it is not recommended to be directly used in input scripts.

The argument *model* is the name of the KIM PM. For potentials archived in OpenKIM this is the extended KIM ID (see *kim command* for details). LAMMPS can invoke any KIM PM, however there can be incompatibilities (for example due to unit matching issues). In the event of an incompatibility, the code will terminate with an error message. Check both the LAMMPS and KIM log files for details.

Only a single *pair_coeff* command is used with the *kim* style, which specifies the mapping of LAMMPS atom types to the species supported by the KIM PM. This is done by specifying *N* additional arguments after the * * in the *pair_coeff* command, where *N* is the number of LAMMPS atom types:

- N element names = mapping of KIM elements to atom types

For example, consider a KIM PM that supports Si and C species. If the LAMMPS simulation has four atom types, where the first three are Si, and the fourth is C, the following *pair_coeff* command would be used:

```
pair_coeff * * Si Si Si C
```

The first two arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1, 2, and 3 to Si as defined within KIM PM. The final C argument maps LAMMPS atom type 4 to C.

In addition to the usual LAMMPS error messages, the KIM library itself may generate errors, which should be printed to the screen. In this case it is also useful to check the *kim.log* file for additional error information. The file *kim.log* should be generated in the same directory where LAMMPS is running.

To download, build, and install the KIM library on your system, see the *lib/kim/README* file. Once you have done this and built LAMMPS with the KIM package installed you can run the example input scripts in *examples/kim*.

4.136.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since KIM stores the potential parameters. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.136.5 Restrictions

This pair style is part of the KIM package. See details on restrictions in *kim command*.

This current version of pair_style kim is compatible with the kim-api package version 2.0.0 and higher.

4.136.6 Related commands

pair_coeff, *kim command*

4.136.7 Default

none

4.137 pair_style kolmogorov/crespi/full command

4.137.1 Syntax

```
pair_style hybrid/overlay kolmogorov/crespi/full cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.137.2 Examples

```
pair_style hybrid/overlay kolmogorov/crespi/full 20.0 0
pair_coeff * * none
pair_coeff * * kolmogorov/crespi/full  CH.KC    C C

pair_style hybrid/overlay rebo kolmogorov/crespi/full 16.0 1
pair_coeff * * rebo                    CH.rebo      C H
pair_coeff * * kolmogorov/crespi/full  CH_taper.KC  C H
```

4.137.3 Description

The *kolmogorov/crespi/full* style computes the Kolmogorov-Crespi (KC) interaction potential as described in *(Kolmogorov)*. No simplification is made,

$$
\begin{aligned}
E =& \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& e^{-\lambda(r_{ij}-z_0)}\left[C + f(\rho_{ij}) + f(\rho_{ji})\right] - A\left(\frac{r_{ij}}{z_0}\right)^{-6} \\
\rho_{ij}^2 =& r_{ij}^2 - (\mathbf{r}_{ij}\cdot\mathbf{n}_i)^2 \\
\rho_{ji}^2 =& r_{ij}^2 - (\mathbf{r}_{ij}\cdot\mathbf{n}_j)^2 \\
f(\rho) =& e^{-(\rho/\delta)^2}\sum_{n=0}^{2} C_{2n}(\rho/\delta)^{2n}
\end{aligned}
$$

It is important to have a sufficiently large cutoff to ensure smooth forces and to include all the pairs to build the neighbor list for calculating the normals. Energies are shifted so that they go continuously to zero at the cutoff assuming that the exponential part of V_{ij} (first term) decays sufficiently fast. This shift is achieved by the last term in the equation for V_{ij} above. This is essential only when the tapper function is turned off. The formula of taper function can be found in pair style *ilp/graphene/hbn*.

Note

This potential (ILP) is intended for interlayer interactions between two different layers of graphene. To perform a realistic simulation, this potential must be used in combination with intralayer potential, such as *AIREBO* or *Tersoff* potential. To keep the intralayer properties unaffected, the interlayer interaction within the same layers should be avoided. Hence, each atom has to have a layer identifier such that atoms residing on the same layer interact via the appropriate intralayer potential and atoms residing on different layers interact via the ILP. Here, the molecule id is chosen as the layer identifier, thus a data file with the "full" atom style is required to use this potential.

The parameter file (e.g. CH.KC), is intended for use with *metal* *units*, with energies in meV. Two additional parameters, *S*, and *rcut* are included in the parameter file. *S* is designed to facilitate scaling of energies. *rcut* is designed to build the neighbor list for calculating the normals for each atom pair.

Note

Two new sets of parameters of KC potential for hydrocarbons, CH.KC (without the taper function) and CH_taper.KC (with the taper function) are presented in *(Ouyang1)*. The energy for the KC potential with the taper function goes continuously to zero at the cutoff. The parameters in both CH.KC and CH_taper.KC provide a good description in both short- and long-range interaction regimes. While the original parameters (CC.KC) published in *(Kolmogorov)* are only suitable for long-range interaction regime. This feature is essential for simulations in high pressure regime (i.e., the interlayer distance is smaller than the equilibrium distance). The benchmark tests and comparison of these parameters can be found in *(Ouyang1)* and *(Ouyang2)*.

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using pair_style *none*.

This pair style tallies a breakdown of the total interlayer potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 2. The 2 values correspond to the following sub-categories:

1. *E_vdW* = vdW (attractive) energy
2. *E_Rep* = Repulsive energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair kolmogorov/crespi/full
variable Evdw  equal c_0[1]
variable Erep  equal c_0[2]
thermo_style custom step temp epair v_Erep v_Evdw
```

4.137.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.137.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the newton setting to be *on* for pair interactions.

The CH.KC potential file provided with LAMMPS (see the potentials folder) is parameterized for metal units. You can use this pair style with any LAMMPS units, but you would need to create your own custom CH.KC potential file with all coefficients converted to the appropriate units.

4.137.6 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style pair_lebedeva_z*, *pair_style kolmogorov/crespi/z*, *pair_style ilp/graphene/hbn*.

4.137.7 Default

tap_flag = 0

(Kolmogorov) A. N. Kolmogorov, V. H. Crespi, Phys. Rev. B 71, 235415 (2005)

(Ouyang1) W. Ouyang, D. Mandelli, M. Urbakh and O. Hod, Nano Lett. 18, 6009-6016 (2018).

(Ouyang2) W. Ouyang et al., J. Chem. Theory Comput. 16(1), 666-676 (2020).

4.138 pair_style kolmogorov/crespi/z command

4.138.1 Syntax

```
pair_style [hybrid/overlay ...] kolmogorov/crespi/z cutoff
```

4.138.2 Examples

```
pair_style hybrid/overlay kolmogorov/crespi/z 20.0
pair_coeff * * none
pair_coeff 1 2 kolmogorov/crespi/z  CC.KC    C C

pair_style hybrid/overlay rebo kolmogorov/crespi/z 14.0
pair_coeff * * rebo                 CH.rebo     C C
pair_coeff 1 2 kolmogorov/crespi/z  CC.KC       C C
```

4.138.3 Description

The *kolmogorov/crespi/z* style computes the Kolmogorov-Crespi interaction potential as described in *(Kolmogorov)*. An important simplification is made, which is to take all normals along the z-axis.

$$
\begin{aligned}
E =& \frac{1}{2} \sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& e^{-\lambda (r_{ij} - z_0)} \left[C + f(\rho_{ij}) + f(\rho_{ji}) \right] - A \left(\frac{r_{ij}}{z_0} \right)^{-6} + A \left(\frac{\text{cutoff}}{z_0} \right)^{-6} \\
\rho_{ij}^2 =& \rho_{ji}^2 = x_{ij}^2 + y_{ij}^2 \qquad\qquad (\mathbf{n}_i \equiv \hat{\mathbf{z}}) \\
f(\rho) =& e^{-(\rho/\delta)^2} \sum_{n=0}^{2} C_{2n} \left(\rho / \delta \right)^{2n}
\end{aligned}
$$

It is important to have a sufficiently large cutoff to ensure smooth forces. Energies are shifted so that they go continuously to zero at the cutoff assuming that the exponential part of V_{ij} (first term) decays sufficiently fast. This shift is achieved by the last term in the equation for V_{ij} above.

This potential is intended for interactions between two layers of graphene. Therefore, to avoid interaction between layers in multi-layered materials, each layer should have a separate atom type and interactions should only be computed between atom types of neighboring layers.

The parameter file (e.g. CC.KC), is intended for use with metal *units*, with energies in meV. An additional parameter, *S*, is available to facilitate scaling of energies in accordance with *(vanWijk)*.

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using pair_style *none*.

4.138.4 Restrictions

This fix is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.138.5 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style ilp/graphene/hbn*. *pair_style kolmogorov/crespi/full*, *pair_style lebedeva/z*

4.138.6 Default

none

(Kolmogorov) A. N. Kolmogorov, V. H. Crespi, Phys. Rev. B 71, 235415 (2005)

(vanWijk) M. M. van Wijk, A. Schuring, M. I. Katsnelson, and A. Fasolino, Physical Review Letters, 113, 135504 (2014)

4.139 pair_style lcbop command

4.139.1 Syntax

```
pair_style lcbop
```

4.139.2 Examples

```
pair_style lcbop
pair_coeff * * ../potentials/C.lcbop C
```

4.139.3 Description

The *lcbop* pair style computes the long-range bond-order potential for carbon (LCBOP) of *(Los and Fasolino)*. See section II in that paper for the analytic equations associated with the potential.

Only a single pair_coeff command is used with the *lcbop* style which specifies an LCBOP potential file with parameters for specific elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of LCBOP elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, if your LAMMPS simulation has 4 atom types and you want the first 3 to be C you would use the following pair_coeff command:

```
pair_coeff * * C.lcbop C C C NULL
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first C argument maps LAMMPS atom type 1 to the C element in the LCBOP file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *lcbop* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The parameters/coefficients for the LCBOP potential as applied to C are listed in the C.lcbop file to agree with the original *(Los and Fasolino)* paper. Thus the parameters are specific to this potential and the way it was fit, so modifying the file should be done carefully.

4.139.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.139.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair potential requires the *newton* setting to be "on" for pair interactions.

The C.lcbop potential file provided with LAMMPS (see the potentials directory) is parameterized for *metal units*. You can use the LCBOP potential with any LAMMPS units, but you would need to create your own LCBOP potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.139.6 Related commands

pair_airebo, *pair_coeff*

4.139.7 Default

none

(Los and Fasolino) J. H. Los and A. Fasolino, Phys. Rev. B 68, 024107 (2003).

4.140 pair_style lebedeva/z command

4.140.1 Syntax

```
pair_style [hybrid/overlay ...] lebedeva/z cutoff
```

4.140.2 Examples

```
pair_style hybrid/overlay lebedeva/z 20.0
pair_coeff * * none
pair_coeff 1 2 lebedeva/z  CC.Lebedeva   C C

pair_style hybrid/overlay rebo lebedeva/z 14.0
pair_coeff * * rebo        CH.rebo       C C
pair_coeff 1 2 lebedeva/z  CC.Lebedeva   C C
```

4.140.3 Description

The *lebedeva/z* pair style computes the Lebedeva interaction potential as described in *(Lebedeva1)* and *(Lebedeva2)*. An important simplification is made, which is to take all normals along the z-axis.

The Lebedeva potential is intended for the description of the interlayer interaction between graphene layers. To perform a realistic simulation, this potential must be used in combination with an intralayer potential such as *AIREBO* or *Tersoff* facilitated by using pair style *hybrid/overlay*. To keep the intralayer properties unaffected, the interlayer interaction within the same layers should be avoided. This can be achieved by assigning different atom types to atoms of different layers (e.g. 1 and 2 in the examples above).

Other interactions can be set to zero using pair_style *none*.

$$
\begin{aligned}
E =& \frac{1}{2} \sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& Be^{-\alpha(r_{ij} - z_0)} \\
& + C(1 + D_1 \rho_{ij}^2 + D_2 \rho_{ij}^4) e^{-\lambda_1 \rho_{ij}^2} e^{-\lambda_2 (z_{ij}^2 - z_0^2)} \\
& - A \left(\frac{z_0}{r_i j} \right)^6 + A \left(\frac{z_0}{r_c} \right)^6 \\
\rho_{ij}^2 =& x_{ij}^2 + y_{ij}^2 \qquad (\mathbf{n_i} \equiv \hat{\mathbf{z}})
\end{aligned}
$$

It is important to have a sufficiently large cutoff to ensure smooth forces. Energies are shifted so that they go continuously to zero at the cutoff assuming that the exponential part of V_{ij} (first term) decays sufficiently fast. This shift is achieved by the last term in the equation for V_{ij} above.

The provided parameter file (CC.Lebedeva) contains two sets of parameters.

- The first set (element name "C") is suitable for normal conditions and is taken from *(Popov1)*
- The second set (element name "C1") is suitable for high-pressure conditions and is taken from *(Koziol1)*

Both sets contain an additional parameter, *S*, that can be used to facilitate scaling of energies and is set to 1.0 by default.

4.140.4 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.140.5 Related commands

pair_coeff, *pair_style none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style ilp/graphene/hbd*, *pair_style kolmogorov/crespi/z*, *pair_style kolmogorov/crespi/full*.

4.140.6 Default

none

(Lebedeva1) I. V. Lebedeva, A. A. Knizhnik, A. M. Popov, Y. E. Lozovik, B. V. Potapkin, Phys. Rev. B, 84, 245437 (2011)

(Lebedeva2) I. V. Lebedeva, A. A. Knizhnik, A. M. Popov, Y. E. Lozovik, B. V. Potapkin, Physica E: 44, 949-954 (2012)

(Popov1) A.M. Popov, I. V. Lebedeva, A. A. Knizhnik, Y. E. Lozovik and B. V. Potapkin, Chem. Phys. Lett. 536, 82-86 (2012).

(Koziol1) Z. Koziol, G. Gawlik and J. Jagielski, Chinese Phys. B 28, 096101 (2019).

4.141 pair_style lepton command

Accelerator Variants: *lepton/omp*, *lepton/coul/comp*, *lepton/sphere/comp*

4.141.1 Syntax

```
pair_style style args
```

- style = *lepton* or *lepton/coul* or *lepton/sphere*
- args = list of arguments for a particular style

lepton args = cutoff
  cutoff = global cutoff for the interactions (distance units)
lepton/coul args = cutoff keyword
  cutoff = global cutoff for the interactions (distance units)
  zero or more keywords may be appended
  keyword = ewald or pppm or msm or dispersion or tip4p
lepton/sphere args = cutoff
  cutoff = global cutoff for the interactions (distance units)

4.141.2 Examples

```
pair_style lepton 2.5

pair_coeff  * *  "k*((r-r0)^2*step(r0-r)); k=200; r0=1.5" 2.0
pair_coeff  1 2  "4.0*eps*((sig/r)^12 - (sig/r)^6);eps=1.0;sig=1.0" 1.12246204830937
pair_coeff  2 2  "eps*(2.0*(sig/r)^9 - 3.0*(sig/r)^6);eps=1.0;sig=1.0"
pair_coeff  1 3  "zbl(13,6,r)"
pair_coeff  3 3  "(1.0-switch)*zbl(6,6,r)-switch*4.0*eps*((sig/r)^6);switch=0.5*(tanh(10.0*(r-sig))+1.0);
 ↪eps=0.05;sig=3.20723"

pair_style lepton/coul 2.5
pair_coeff 1 1 "qi*qj/r" 4.0
pair_coeff 1 2 "lj+coul; lj=4.0*eps*((sig/r)^12 - (sig/r)^6); eps=1.0; sig=1.0; coul=qi*qj/r"

pair_style lepton/coul 2.5 pppm
kspace_style pppm 1.0e-4
pair_coeff 1 1 "qi*qj/r*erfc(alpha*r); alpha=1.067"

pair_style lepton/sphere 2.5
pair_coeff 1 * "k*((r-r0)^2*step(r0-r)); k=200; r0=radi+radj"
pair_coeff 2 2 "4.0*eps*((sig/r)^12 - (sig/r)^6); eps=1.0; sig=2.0*sqrt(radi*radj)"
```

4.141.3 Description

Added in version 8Feb2023: added pair styles *lepton* and *lepton/coul*

Changed in version 15Jun2023: added pair style *lepton/sphere*

Pair styles *lepton*, *lepton/coul*, *lepton/sphere* compute pairwise interactions between particles which depend on the distance and have a cutoff. The potential function must be provided as an expression string using "r" as the distance variable. With pair style *lepton/coul* one may additionally reference the charges of the two atoms of the pair with "qi" and "qj", respectively. With pair style *lepton/sphere* one may instead reference the radii of the two atoms of the pair with "radi" and "radj", respectively; this is half of the diameter that can be set in *data files* or the *set command*.

Note that further constants in the expressions can be defined in the same string as additional expressions separated by semicolons as shown in the examples above.

The expression *"200.0*(r-1.5)^2"* represents a harmonic potential around the pairwise distance r_0 of 1.5 distance units and a force constant K of 200.0 energy units:

$$U_{ij} = K(r - r_0)^2$$

The expression *"qi*qj/r"* represents a regular Coulombic potential with cutoff:

$$U_{ij} = \frac{Cq_iq_j}{\varepsilon r} \qquad r < r_c$$

The expression *"200.0*(r-(radi+radj)^2"* represents a harmonic potential that has the equilibrium distance chosen so that the radii of the two atoms touch:

$$U_{ij} = K(r - (r_i + r_j))^2$$

The Lepton library, that the *lepton* pair style interfaces with, evaluates this expression string at run time to compute the pairwise energy. It also creates an analytical representation of the first derivative of this expression with respect to "r" and then uses that to compute the force between the pairs of particles within the given cutoff.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- Lepton expression (energy units)
- cutoff (distance units)

The Lepton expression must be either enclosed in quotes or must not contain any whitespace so that LAMMPS recognizes it as a single keyword. More on valid Lepton expressions below. The last coefficient is optional; it allows to set the cutoff for a pair of atom types to a different value than the global cutoff.

For pair style *lepton* only the "lj" values of the *special_bonds* settings apply in case the interacting pair is also connected with a bond. The potential energy will *only* be added to the "evdwl" property.

For pair style *lepton/coul* only the "coul" values of the *special_bonds* settings apply in case the interacting pair is also connected with a bond. The potential energy will *only* be added to the "ecoul" property.

For pair style *lepton/sphere* only the "lj" values of the *special_bonds* settings apply in case the interacting pair is also connected with a bond. The potential energy will *only* be added to the "evdwl" property.

In addition to the functions listed below, both pair styles support in addition a custom "zbl(zi,zj,r)" function which computes the Ziegler-Biersack-Littmark (ZBL) screened nuclear repulsion for describing high-energy collisions between atoms. For details of the function please see the documentation for *pair style zbl*. The arguments of the function are the atomic numbers of atom i (zi), atom j (zj) and the distance r. Please see the examples above.

4.141.4 Lepton expression syntax and features

Lepton supports the following operators in expressions:

+	Add	-	Subtract	*	Multiply	/	Divide	^	Power

The following mathematical functions are available:

sqrt(x)	Square root	exp(x)	Exponential
log(x)	Natural logarithm	sin(x)	Sine (angle in radians)
cos(x)	Cosine (angle in radians)	sec(x)	Secant (angle in radians)
csc(x)	Cosecant (angle in radians)	tan(x)	Tangent (angle in radians)
cot(x)	Cotangent (angle in radians)	asin(x)	Inverse sine (in radians)
acos(x)	Inverse cosine (in radians)	atan(x)	Inverse tangent (in radians)
sinh(x)	Hyperbolic sine	cosh(x)	Hyperbolic cosine
tanh(x)	Hyperbolic tangent	erf(x)	Error function
erfc(x)	Complementary Error function	abs(x)	Absolute value
min(x,y)	Minimum of two values	max(x,y)	Maximum of two values
delta(x)	delta(x) is 1 for $x = 0$, otherwise 0	step(x)	step(x) is 0 for $x < 0$, otherwise 1

Numbers may be given in either decimal or exponential form. All of the following are valid numbers: *5*, *-3.1*, *1e6*, and *3.12e-2*.

As an extension to the standard Lepton syntax, it is also possible to use LAMMPS *variables* in the format "v_name". Before evaluating the expression, "v_name" will be replaced with the value of the variable "name". This is compatible with all kinds of scalar variables, but not with vectors, arrays, local, or per-atom variables. If necessary, a custom scalar variable needs to be defined that can access the desired (single) item from a non-scalar variable. As an example, the following lines will instruct LAMMPS to ramp the force constant for a harmonic bond from 100.0 to 200.0 during the next run:

```
variable fconst equal ramp(100.0, 200)
bond_style lepton
bond_coeff 1 1.5 "v_fconst * (r^2)"
```

An expression may be followed by definitions for intermediate values that appear in the expression. A semicolon ";" is used as a delimiter between value definitions. For example, the expression:

```
a^2+a*b+b^2; a=a1+a2; b=b1+b2
```

is exactly equivalent to

```
(a1+a2)^2+(a1+a2)*(b1+b2)+(b1+b2)^2
```

The definition of an intermediate value may itself involve other intermediate values. Whitespace and quotation characters ('' and "") are ignored. All uses of a value must appear *before* that value's definition. For efficiency reasons, the expression string is parsed, optimized, and then stored in an internal, pre-parsed representation for evaluation.

Evaluating a Lepton expression is typically between 2.5 and 5 times slower than the corresponding compiled and optimized C++ code. If additional speed or GPU acceleration (via GPU or KOKKOS) is required, the interaction can be represented as a table. Suitable table files can be created either internally using the *pair_write* or *bond_write* command or through the Python scripts in the *tools/tabulate* folder.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.141.5 Mixing, shift, table, tail correction, restart, rRESPA info

Pair styles *lepton*, *lepton/coul*, and *lepton/sphere* do not support mixing. Thus, expressions for *all* I,J pairs must be specified explicitly.

Only pair style *lepton* supports the *pair_modify shift* option for shifting the potential energy of the pair interaction so that it is 0 at the cutoff, pair styles *lepton/coul* and *lepton/sphere* do *not*.

The *pair_modify table* options are not relevant for the these pair styles.

These pair styles do not support the *pair_modify tail* option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.141.6 Restrictions

These pair styles are part of the LEPTON package and only enabled if LAMMPS was built with this package. See the *Build package* page for more info.

Pair style *lepton/coul* requires that atom atoms have a charge property, e.g. via *atom_style charge*.

Pair style *lepton/sphere* requires that atom atoms have a radius property, e.g. via *atom_style sphere*.

4.141.7 Related commands

pair_coeff, *pair_style python*, *pair_style table*, *pair_write*

4.141.8 Default

none

4.142 pair_style line/lj command

4.142.1 Syntax

```
pair_style line/lj cutoff
```

cutoff = global cutoff for interactions (distance units)

4.142.2 Examples

```
pair_style line/lj 3.0
pair_coeff * * 1.0 1.0 1.0 0.8 1.12
pair_coeff 1 2 1.0 2.0 1.0 1.5 1.12 5.0
pair_coeff 1 2 1.0 0.0 1.0 1.0 2.5
```

4.142.3 Description

Style *line/lj* treats particles which are line segments as a set of small spherical particles that tile the line segment length as explained below. Interactions between two line segments, each with N1 and N2 spherical particles, are calculated as the pairwise sum of N1*N2 Lennard-Jones interactions. Interactions between a line segment with N spherical particles and a point particle are treated as the pairwise sum of N Lennard-Jones interactions. See the *pair_style lj/cut* page for the definition of Lennard-Jones interactions.

The set of non-overlapping spherical sub-particles that represent a line segment are generated in the following manner. Their size is a function of the line segment length and the specified sub-particle size for that particle type. If a line segment has a length L and is of type I, then the number of spheres N that represent the segment is calculated as N = L/sizeI, rounded up to an integer value. Thus if L is not evenly divisible by sizeI, N is incremented to include one extra sphere. The centers of the spheres are spaced equally along the line segment. Imagine N+1 equally-space points, which include the 2 end points of the segment. The sphere centers are halfway between each pair of points.

The LJ interaction between 2 spheres on different line segments (or a sphere on a line segment and a point particles) is computed with sub-particle ε, σ, and *cutoff* values that are set by the pair_coeff command, as described below. If the distance between the 2 spheres is greater than the sub-particle cutoff, there is no interaction. This means that some pairs of sub-particles on 2 line segments may interact, but others may not.

For purposes of creating the neighbor list for pairs of interacting line segments or lines/point particles, a regular particle-particle cutoff is used, as defined by the *cutoff* setting above in the pair_style command or overridden with an optional argument in the pair_coeff command for a type pair as discussed below. The distance between the centers of 2 line segments, or the center of a line segment and a point particle, must be less than this distance (plus the neighbor skin; see the *neighbor* command), for the pair of particles to be included in the neighbor list.

Note

This means that a too-short value for the *cutoff* setting can exclude a pair of particles from the neighbor list even if pairs of their sub-particle spheres would interact, based on the sub-particle cutoff specified in the pair_coeff command. E.g. sub-particles at the ends of the line segments that are close to each other. Which may not be what

you want, since it means the ends of 2 line segments could pass through each other. It is up to you to specify a *cutoff* setting that is consistent with the length of the line segments you are using and the sub-particle cutoff settings.

For style *line/lj*, the following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- sizeI (distance units)
- sizeJ (distance units)
- ε (energy units)
- σ (distance units)
- subcutoff (distance units)
- cutoff (distance units)

The *sizeI* and *sizeJ* coefficients are the sub-particle sizes for line particles of type I and type J. They are used to define the N sub-particles per segment as described above. These coefficients are actually stored on a per-type basis. Thus if there are multiple pair_coeff commands that involve type I, as either the first or second atom type, you should use consistent values for sizeI or sizeJ in all of them. If you do not do this, the last value specified for sizeI will apply to all segments of type I. If typeI or typeJ refers to point particles, the corresponding sizeI or sizeJ is ignored; it can be set to 0.0.

The ε, σ, and *subcutoff* coefficients are used to compute an LJ interactions between a pair of sub-particles on 2 line segments (of type I and J), or between a sub particle/point particle pair. As discussed above, the *subcutoff* and *cutoff* params are different. The latter is only used for building the neighbor list when the distance between centers of two line segments or one segment and a point particle is calculated.

The *cutoff* coefficient is optional. If not specified, the global cutoff is used.

4.142.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, coefficients must be specified. No default mixing rules are used.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.142.5 Restrictions

This style is part of the ASPHERE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Defining particles to be line segments so they participate in line/line or line/particle interactions requires the use the *atom_style line* command.

4.142.6 Related commands

pair_coeff, *pair_style tri/lj*

4.142.7 Default

none

4.143 pair_style list command

4.143.1 Syntax

```
pair_style list listfile cutoff keyword
```

- listfile = name of file with list of pairwise interactions
- cutoff = global cutoff (distance units)
- keyword = optional flag *nocheck* or *check* (default is *check*)

4.143.2 Examples

```
pair_style list restraints.txt 200.0
pair_coeff * *

pair_style hybrid/overlay lj/cut 1.1225 list pair_list.txt 300.0
pair_coeff * * lj/cut 1.0 1.0
pair_coeff 3* 3* list
```

4.143.3 Description

Style *list* computes interactions between explicitly listed pairs of atoms with the option to select functional form and parameters for each individual pair. Because the parameters are set in the list file, the pair_coeff command has no parameters (but still needs to be provided). The *check* and *nocheck* keywords enable/disable tests that checks whether all listed pairs of atom IDs were present and the interactions computed. If *nocheck* is set and either atom ID is not present, the interaction is skipped.

This pair style can be thought of as a hybrid between bonded, non-bonded, and restraint interactions. It will typically be used as an additional interaction within the *hybrid/overlay* pair style. It currently supports three interaction styles: a 12-6 Lennard-Jones, a Morse and a harmonic potential.

The format of the list file is as follows:

- one line per pair of atoms
- empty lines will be ignored
- comment text starts with a '#' character
- line syntax: *ID1 ID2 style coeffs cutoff*

```
ID1 = atom ID of first atom
ID2 = atom ID of second atom
style = style of interaction
coeffs = list of coeffs
cutoff = cutoff for interaction (optional)
```

The cutoff parameter is optional for all but the *quartic* interactions. If it is not specified, the global cutoff is used.

Here is an example file:

```
# this is a comment

15 259 lj126     1.0 1.0      50.0
15 603 morse     10.0 1.2 2.0  10.0 # and another comment
18 470 harmonic 50.0 1.2       5.0
19 332 quartic  10.0 5.0 -1.2 1.2
```

The style *lj126* computes pairwise interactions with the formula

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

and the coefficients:

- ε (energy units)
- σ (distance units)

The style *morse* computes pairwise interactions with the formula

$$E = D_0 \left[1 - e^{-\alpha(r-r_0)}\right]^2 \qquad r < r_c$$

and the coefficients:

- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)

The style *harmonic* computes pairwise interactions with the formula

$$E = K(r - r_0)^2 \qquad r < r_c$$

and the coefficients:

- K (energy units)
- r_0 (distance units)

Note that the usual 1/2 factor is included in K.

The style *quartic* computes pairwise interactions with the formula

$$E = K(r - r_0)^2(r - r_0 - b_1)(r - r_0 - b_2) \qquad r < r_c$$

and the coefficients:

- K (energy units)
- r_0 (distance units)

- b_1 (distance units)
- b_2 (distance units)

4.143.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing since all parameters are explicit for each pair.

The *pair_modify* shift option is supported by this pair style.

The *pair_modify* table and tail options are not relevant for this pair style.

This pair style does not write its information to *binary restart files*, so pair_style and pair_coeff commands need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.143.5 Restrictions

This pair style does not use a neighbor list and instead identifies atoms by their IDs. This has two consequences: 1) The cutoff has to be chosen sufficiently large, so that the second atom of a pair has to be a ghost atom on the same node on which the first atom is local; otherwise the interaction will be skipped. You can use the *check* option to detect, if interactions are missing. 2) Unlike other pair styles in LAMMPS, an atom I will not interact with multiple images of atom J (assuming the images are within the cutoff distance), but only with the closest image.

This style is part of the MISC package. It is only enabled if LAMMPS is build with that package. See the *Build package* page on for more info.

4.143.6 Related commands

pair_coeff , *pair_style hybrid/overlay*, *pair_style lj/cut*, *bond_style morse*, *bond_style harmonic bond_style quartic*

4.143.7 Default

none

4.144 pair_style lj/cut command

Accelerator Variants: *lj/cut/gpu*, *lj/cut/intel*, *lj/cut/kk*, *lj/cut/opt*, *lj/cut/omp*

4.144.1 Syntax

```
pair_style style args
```

- style = *lj/cut*
- args = list of arguments for a particular style

lj/cut args = cutoff
  cutoff = global cutoff for Lennard Jones interactions (distance units)

4.144.2 Examples

```
pair_style lj/cut 2.5
pair_coeff * * 1 1
pair_coeff 1 1 1 1.1 2.8
```

4.144.3 Description

The *lj/cut* styles compute the standard 12/6 Lennard-Jones potential, given by

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

r_c is the cutoff.

See the *lj/cut/coul* styles to add a Coulombic pairwise interaction and the *lj/cut/tip4p* styles to add the TIP4P water model.

4.144.4 Coefficients

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- LJ cutoff (distance units)

The last coefficient is optional. If not specified, the global LJ cutoff specified in the pair_style command is used.

Note that σ is defined in the LJ formula as the zero-crossing distance for the potential, *not* as the energy minimum at $r_0 = 2^{\frac{1}{6}}\sigma$. The _same_ potential function becomes:

$$E = \varepsilon \left[\left(\frac{r_0}{r}\right)^{12} - 2\left(\frac{r_0}{r}\right)^{6} \right] \qquad r < r_c$$

When using the minimum as reference width. In the literature both formulations are used, but the describe the same potential, only the σ value must be computed by $\sigma = r_0/2^{\frac{1}{6}}$ for use with LAMMPS, if this latter formulation is used.

A version of these styles with a soft core, *lj/cut/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.144.5 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

All of the *lj/cut* pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

All of the *lj/cut* pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure for the Lennard-Jones portion of the pair interaction.

All of the *lj/cut* pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The *lj/cut* pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. The other styles only support the *pair* keyword of run_style respa. See the *run_style* command for details.

4.144.6 Related commands

- *pair_coeff*
- *pair_style lj/cut/coul/cut*
- *pair_style lj/cut/coul/debye*
- *pair_style lj/cut/coul/dsf*
- *pair_style lj/cut/coul/long*
- *pair_style lj/cut/coul/msm*
- *pair_style lj/cut/coul/wolf*
- *pair_style lj/cut/tip4p/cut*
- *pair_style lj/cut/tip4p/long*

4.144.7 Default

none

4.145 pair_style lj96/cut command

Accelerator Variants: *lj96/cut/gpu*, *lj96/cut/omp*

4.145.1 Syntax

```
pair_style lj96/cut cutoff
```

- cutoff = global cutoff for lj96/cut interactions (distance units)

4.145.2 Examples

```
pair_style lj96/cut 2.5
pair_coeff * * 1.0 1.0 4.0
pair_coeff 1 1 1.0 1.0
```

4.145.3 Description

The *lj96/cut* style compute a 9/6 Lennard-Jones potential, instead of the standard 12/6 potential, given by

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^9 - \left(\frac{\sigma}{r}\right)^6 \right] \qquad r < r_c$$

r_c is the cutoff.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global LJ cutoff specified in the pair_style command is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.145.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style supports the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure of the pair interaction.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style supports the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. See the *run_style* command for details.

4.145.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.145.6 Related commands

pair_coeff

4.145.7 Default

none

4.146 pair_style lj/cubic command

Accelerator Variants: *lj/cubic/gpu*, *lj/cubic/omp*

4.146.1 Syntax

```
pair_style lj/cubic
```

4.146.2 Examples

```
pair_style lj/cubic
pair_coeff * * 1.0 0.8908987
```

4.146.3 Description

The *lj/cubic* style computes a truncated LJ interaction potential whose energy and force are continuous everywhere. Inside the inflection point the interaction is identical to the standard 12/6 *Lennard-Jones* potential. The LJ function outside the inflection point is replaced with a cubic function of distance. The energy, force, and second derivative are continuous at the inflection point. The cubic coefficient A3 is chosen so that both energy and force go to zero at the cutoff distance. Outside the cutoff distance the energy and force are zero.

$$
\begin{aligned}
E &= u_{LJ}(r) \qquad r \le r_s \\
&= u_{LJ}(r_s) + (r - r_s)u'_{LJ}(r_s) - \frac{1}{6}A_3(r - r_s)^3 \qquad r_s < r \le r_c \\
&= 0 \qquad r > r_c
\end{aligned}
$$

The location of the inflection point r_s is defined by the LJ diameter, $r_s/\sigma = (26/7)^{1/6}$. The cutoff distance is defined by $r_c/r_s = 67/48$ or $r_c/\sigma = 1.737...$ The analytic expression for the the cubic coefficient $A_3 r_{min}^3/\varepsilon = 27.93...$ is given in the paper by Holian and Ravelo *(Holian)*.

This potential is commonly used to study the shock mechanics of FCC solids, as in Ravelo et al. *(Ravelo)*.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)

Note that σ is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum, which is located at $r_{min} = 2^{\frac{1}{6}}\sigma$. In the above example, $\sigma = 0.8908987$, so $r_{min} = 1.0$.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.146.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

The lj/cubic pair style does not support the *pair_modify* shift option, since pair interaction is already smoothed to 0.0 at the cutoff.

The *pair_modify* table option is not relevant for this pair style.

The lj/cubic pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since there are no corrections for a potential that goes to 0.0 at the cutoff.

The lj/cubic pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The lj/cubic pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.146.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.146.6 Related commands

pair_coeff

4.146.7 Default

none

(Holian) Holian and Ravelo, Phys Rev B, 51, 11275 (1995).

(Ravelo) Ravelo, Holian, Germann and Lomdahl, Phys Rev B, 70, 014103 (2004).

4.147 pair_style lj/cut/coul/cut command

Accelerator Variants: *lj/cut/coul/cut/gpu*, *lj/cut/coul/cut/kk*, *lj/cut/coul/cut/omp*

4.148 pair_style lj/cut/coul/debye command

Accelerator Variants: *lj/cut/coul/debye/gpu*, *lj/cut/coul/debye/kk*, *lj/cut/coul/debye/omp*

4.149 pair_style lj/cut/coul/dsf command

Accelerator Variants: *lj/cut/coul/dsf/gpu*, *lj/cut/coul/dsf/kk*, *lj/cut/coul/dsf/omp*

4.150 pair_style lj/cut/coul/long command

Accelerator Variants: *lj/cut/coul/long/gpu*, *lj/cut/coul/long/kk*, *lj/cut/coul/long/intel*, *lj/cut/coul/long/opt*, *lj/cut/coul/long/omp*

4.151 pair_style lj/cut/coul/msm command

Accelerator Variants: *lj/cut/coul/msm/gpu*, *lj/cut/coul/msm/omp*

4.152 pair_style lj/cut/coul/wolf command

Accelerator Variants: *lj/cut/coul/wolf/omp*

4.152.1 Syntax

```
pair_style style args
```

- style = *lj/cut/coul/cut* or *lj/cut/coul/debye* or *lj/cut/coul/dsf* or *lj/cut/coul/long lj/cut/coul/msm* or *lj/cut/coul/wolf*
- args = list of arguments for a particular style

lj/cut/coul/cut args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/coul/debye args = kappa cutoff (cutoff2)
  kappa = inverse of the Debye length (inverse distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/coul/dsf args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (distance units)
lj/cut/coul/long args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/coul/msm args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/coul/wolf args = alpha cutoff (cutoff2)
  alpha = damping parameter (inverse distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 2 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.152.2 Examples

```
pair_style lj/cut/coul/cut 10.0
pair_style lj/cut/coul/cut 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0
pair_coeff 1 1 100.0 3.5 9.0 9.0

pair_style lj/cut/coul/debye 1.5 3.0
pair_style lj/cut/coul/debye 1.5 2.5 5.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.0 1.5 2.5
pair_coeff 1 1 1.0 1.5 2.5 5.0

pair_style lj/cut/coul/dsf 0.05 2.5 10.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.0 1.0 2.5

pair_style lj/cut/coul/long 10.0
pair_style lj/cut/coul/long 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/cut/coul/msm 10.0
pair_style lj/cut/coul/msm 10.0 8.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/cut/coul/wolf 0.2 5. 10.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.0 1.0 2.5
```

4.152.3 Description

The *lj/cut/coul* styles compute the standard 12/6 Lennard-Jones potential, given by

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

r_c is the cutoff.

Style *lj/cut/coul/cut* adds a Coulombic pairwise interaction given by

$$E = \frac{C q_i q_j}{\varepsilon r} \qquad r < r_c$$

where C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, and ε is the dielectric constant which can be set by the *dielectric* command. If one cutoff is specified in the pair_style command, it is used for both the LJ and Coulombic terms. If two cutoffs are specified, they are used as cutoffs for the LJ and Coulombic terms respectively.

Style *lj/cut/coul/debye* adds an additional exp() damping factor to the Coulombic term, given by

$$E = \frac{C q_i q_j}{\varepsilon r} \exp(-\kappa r) \qquad r < r_c$$

where κ is the inverse of the Debye length. This potential is another way to mimic the screening effect of a polar solvent.

Style *lj/cut/coul/dsf* computes the Coulombic term via the damped shifted force model described in *Fennell*, given by:

$$E = q_i q_j \left[\frac{\mathrm{erfc}(\alpha r)}{r} - \frac{\mathrm{erfc}(\alpha r_c)}{r_c} + \left(\frac{\mathrm{erfc}(\alpha r_c)}{r_c^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r_c^2)}{r_c} \right) (r - r_c) \right] \qquad r < r_c$$

where α is the damping parameter and erfc() is the complementary error-function. This potential is essentially a short-range, spherically-truncated, charge-neutralized, shifted, pairwise *1/r* summation. The potential is based on Wolf summation, proposed as an alternative to Ewald summation for condensed phase systems where charge screening causes electrostatic interactions to become effectively short-ranged. In order for the electrostatic sum to be absolutely convergent, charge neutralization within the cutoff radius is enforced by shifting the potential through placement of image charges on the cutoff sphere. Convergence can often be improved by setting α to a small non-zero value.

Styles *lj/cut/coul/long* and *lj/cut/coul/msm* compute the same Coulombic interactions as style *lj/cut/coul/cut* except that an additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

Style *lj/cut/coul/wolf* adds a Coulombic pairwise interaction via the Wolf summation method, described in *Wolf*, given by:

$$E_i = \frac{1}{2} \sum_{j \neq i} \frac{q_i q_j \mathrm{erfc}(\alpha r_{ij})}{r_{ij}} + \frac{1}{2} \sum_{j \neq i} \frac{q_i q_j \mathrm{erf}(\alpha r_{ij})}{r_{ij}} \qquad r < r_c$$

where α is the damping parameter, and erfc() is the complementary error-function terms. This potential is essentially a short-range, spherically-truncated, charge-neutralized, shifted, pairwise *1/r* summation. With a manipulation of adding and subtracting a self term (for i = j) to the first and second term on the right-hand-side, respectively, and a small enough α damping parameter, the second term shrinks and the potential becomes a rapidly-converging real-space summation. With a long enough cutoff and small enough α parameter, the energy and forces calculated by the Wolf summation method approach those of the Ewald sum. So it is a means of getting effective long-range interactions with a short-range potential.

4.152.4 Coefficients

For all of the *lj/cut/coul* pair styles, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff1 (distance units)
- cutoff2 (distance units)

Note that σ is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum at $2^{\frac{1}{6}}\sigma$.

The latter 2 coefficients are optional. If not specified, the global LJ and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both LJ and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the LJ and Coulombic cutoffs for this type pair.

For *lj/cut/coul/long* and *lj/cut/coul/msm* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

A version of these styles with a soft core, *lj/cut/coul/soft*and *lj/cut/coul/long/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.152.5 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

All of the *lj/cut* pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/cut/coul/long* pair styles support the *pair_modify* table option since they can tabulate the short-range portion of the long-range Coulombic interaction.

All of the *lj/cut* pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure for the Lennard-Jones portion of the pair interaction.

All of the *lj/cut* pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The *lj/cut/coul/long* pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. The other styles only support the *pair* keyword of run_style respa. See the *run_style* command for details.

4.152.6 Restrictions

The *lj/cut/coul/long* and *lj/cut/coul/msm* styles are part of the KSPACE package.

The *lj/cut/coul/debye*, *lj/cut/coul/dsf*, and *lj/cut/coul/wolf* styles are part of the EXTRA-PAIR package.

These styles are only enabled if LAMMPS was built with those respective packages. See the *Build package* page for more info.

4.152.7 Related commands

pair_coeff

4.152.8 Default

none

(Wolf) D. Wolf, P. Keblinski, S. R. Phillpot, J. Eggebrecht, J Chem Phys, 110, 8254 (1999).

(Fennell) C. J. Fennell, J. D. Gezelter, J Chem Phys, 124, 234104 (2006).

4.153 pair_style lj/cut/sphere command

Accelerator Variant: *lj/cut/sphere/omp*

4.153.1 Syntax

```
pair_style style args
```

- style = *lj/cut/sphere*
- args = list of arguments for a particular style

lj/cut/sphere args = cutoff ratio
  cutoff = global cutoff ratio for Lennard Jones interactions (unitless)

4.153.2 Examples

```
pair_style lj/cut/sphere 2.5
pair_coeff * * 1.0
pair_coeff 1 1 1.1 2.8
```

4.153.3 Description

Added in version 15Jun2023.

The *lj/cut/sphere* style compute the standard 12/6 Lennard-Jones potential, given by

$$E = 4\varepsilon \left[\left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^{6} \right] \qquad r < r_c * \sigma_{ij}$$

r_c is the cutoff ratio.

This is the same potential function used by the *lj/cut* pair style, but the σ_{ij} parameter is not set as a per-type parameter via the *pair_coeff command*. Instead it is calculated individually for each pair using the per-atom diameter attribute of *atom_style sphere* for the two atoms as σ_i and σ_j; σ_{ij} is then computed by the mixing rule for pair coefficients as set by the *pair_modify mix* command (defaults to geometric mixing). The cutoff is not specified as a distance, but as ratio that is internally multiplied by σ_{ij} to obtain the actual cutoff for each pair of atoms.

Note that σ_{ij} is defined in the LJ formula above as the zero-crossing distance for the potential, *not* as the energy minimum which is at $2^{\frac{1}{6}}\sigma_{ij}$.

Notes on cutoffs, neighbor lists, and efficiency

If your system is mildly polydisperse, meaning the ratio of the diameter of the largest particle to the smallest is less than 2.0, then the neighbor lists built by the code should be reasonably efficient. Which means they will not contain too many particle pairs that do not interact. However, if your system is highly polydisperse (ratio > 2.0), the neighbor list build and force computations may be inefficient. There are two ways to try and speed up the simulations.

The first is to assign atoms to different atom types so that atoms of each type are similar in size. E.g. if particle diameters range from 1 to 5 use 4 atom types, ensuring atoms of type 1 have diameters from 1.0-2.0, type 2 from 2.0-3.0, etc. This will reduce the number of non-interacting pairs in the neighbor lists and thus reduce the time spent on computing pairwise interactions.

The second is to use the *neighbor multi* command which enabled a different algorithm for building neighbor lists. This will also require that you assign multiple atom types according to diameters, but will in addition use a more efficient size-dependent strategy to construct the neighbor lists and thus reduce the time spent on building neighbor lists.

Here are example input script commands using both ideas for a highly polydisperse system:

```
units           lj
atom_style      sphere
lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      2 box
create_atoms    1 box

# create atoms with random diameters from bimodal distribution
variable switch atom random(0.0,1.0,345634)
variable diam atom (v_switch<0.75)*normal(0.4,0.075,325)+(v_switch>=0.7)*normal(1.2,0.2,453)
set group all diameter v_diam

# assign type 2 to atoms with diameter > 0.6
variable large atom (2.0*radius)>0.6
group large variable large
set group large type 2

pair_style      lj/cut/sphere 2.5
pair_coeff      * * 1.0

neighbor 0.3 multi
```

Using multiple atom types speeds up the calculation for this example by more than a factor of 2, and using the multi-style neighbor list build causes an additional speedup of about 20 percent.

4.153.4 Coefficients

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- LJ cutoff ratio (unitless) (optional)

The last coefficient is optional. If not specified, the global LJ cutoff ratio specified in the *pair_style command* is used.

If a repulsive only LJ interaction is desired, the coefficient for the cutoff ratio should be set to the minimum of the LJ potential using $\$(2.0\hat{}(1.0/6.0))$

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.153.5 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon coefficients and cutoff ratio for the *lj/cut/sphere* pair style can be mixed. The default mixing style is *geometric*. See the *pair_modify command* for details.

The *lj/cut/sphere* pair style supports the *pair_modify shift* option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/cut/sphere* pair style does *not* support the *pair_modify* tail option for adding a long-range tail corrections to the energy and pressure.

The *lj/cut/sphere* pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does *not* support the *inner*, *middle*, *outer* keywords.

4.153.6 Restrictions

The *lj/cut/sphere* pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

The *lj/cut/sphere* pair style does not support the *sixthpower* mixing rule.

4.153.7 Related commands

- *pair_coeff*
- *pair_style lj/cut*
- *pair_style lj/expnd/sphere*

4.153.8 Default

none

4.154 pair_style lj/cut/tip4p/cut command

Accelerator Variants: *lj/cut/tip4p/cut/omp*

4.155 pair_style lj/cut/tip4p/long command

Accelerator Variants: *lj/cut/tip4p/long/gpu*, *lj/cut/tip4p/long/omp*, *lj/cut/tip4p/long/opt*

4.155.1 Syntax

```
pair_style style args
```

- style = *lj/cut/tip4p/cut* or *lj/cut/tip4p/long*
- args = list of arguments for a particular style

lj/cut/tip4p/cut args = otype htype btype atype qdist cutoff (cutoff2)
  otype,htype = atom types (numeric or type label) for TIP4P O and H
  btype,atype = bond and angle types (numeric or type label) for TIP4P waters
  qdist = distance from O atom to massless charge (distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/cut/tip4p/long args = otype htype btype atype qdist cutoff (cutoff2)
  otype,htype = atom types (numeric or type label) for TIP4P O and H
  btype,atype = bond and angle types (numeric or type label) for TIP4P waters
  qdist = distance from O atom to massless charge (distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.155.2 Examples

```
pair_style lj/cut/tip4p/cut 1 2 7 8 0.15 12.0
pair_style lj/cut/tip4p/cut 1 2 7 8 0.15 12.0 10.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/cut/tip4p/long 1 2 7 8 0.15 12.0
pair_style lj/cut/tip4p/long 1 2 7 8 0.15 12.0 10.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/cut/tip4p/long OW HW HW-OW HW-OW-HW 0.15 12.0
labelmap atom 1 OW 2 HW
labelmap bond 1 HW-OW
labelmap angle 1 HW-OW-HW
pair_coeff * * 100.0 3.0
pair_coeff OW OW 100.0 3.5 9.0
```

4.155.3 Description

The *lj/cut/tip4p* styles implement the TIP4P water model of *(Jorgensen)* and similar models, which introduce a massless site M located a short distance away from the oxygen atom along the bisector of the HOH angle. The atomic types of the oxygen and hydrogen atoms, the bond and angle types for OH and HOH interactions, and the distance to the massless charge site are specified as pair_style arguments and are used to identify the TIP4P-like molecules and determine the position of the M site from the positions of the hydrogen and oxygen atoms of the water molecules. The M site location is used for all Coulomb interactions instead of the oxygen atom location, also with all other atom types, while the location of the oxygen atom is used for the Lennard-Jones interactions. Style *lj/cut/tip4p/cut* uses a cutoff for Coulomb interactions; style *lj/cut/tip4p/long* is for use with a long-range Coulombic solver (Ewald or PPPM).

Note

For each TIP4P water molecule in your system, the atom IDs for the O and 2 H atoms must be consecutive, with the O atom first. This is to enable LAMMPS to "find" the 2 H atoms associated with each O atom. For example, if the atom ID of an O atom in a TIP4P water molecule is 500, then its 2 H atoms must have IDs 501 and 502.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

See the *Howto tip4p* page for more information on how to use the TIP4P pair styles and lists of parameters to set. Note that the neighbor list cutoff for Coulomb interactions is effectively extended by a distance 2*qdist when using the TIP4P pair style, to account for the offset distance of the fictitious charges on O atoms in water molecules. Thus it is typically best in an efficiency sense to use a LJ cutoff >= Coulombic cutoff + 2*qdist, to shrink the size of the neighbor list. This leads to slightly larger cost for the long-range calculation, so you can test the trade-off for your model.

The *lj/cut/tip4p* styles compute the standard 12/6 Lennard-Jones potential, given by

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

r_c is the cutoff.

They add Coulombic pairwise interactions given by

$$E = \frac{Cq_iq_j}{\varepsilon r} \qquad r < r_c$$

where C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, and ε is the dielectric constant which can be set by the *dielectric* command. If one cutoff is specified in the pair_style command, it is used for both the LJ and Coulombic terms. If two cutoffs are specified, they are used as cutoffs for the LJ and Coulombic terms respectively.

Style *lj/cut/tip4p/long* compute the same Coulombic interactions as style *lj/cut/tip4p/cut* except that an additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

4.155.4 Coefficients

For all of the *lj/cut* pair styles, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- LJ cutoff (distance units)

Note that σ is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum at $2^{\frac{1}{6}}\sigma$.

The last coefficient is optional. If not specified, the global LJ cutoff specified in the pair_style command is used.

For *lj/cut/tip4p/cut* and *lj/cut/tip4p/long* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

Warning

Because of how these pair styles implement the coulomb interactions by implicitly defining a fourth site for the negative charge of the TIP4P and similar water models, special care must be taken when using these pair styles with other computations that also use charges. Unless they are specially set up to also handle the implicit definition of the 4th site, results are likely incorrect. Example: *compute dipole/chunk*. For the same reason, when using one of these pair styles with *pair_style hybrid*, **all** coulomb interactions should be handled by a single sub-style with TIP4P support. All other instances and styles will "see" the M point charges at the position of the Oxygen atom and thus compute incorrect forces and energies. LAMMPS will print a warning when it detects one of these issues.

A version of these styles with a soft core, *lj/cut/tip4p/long/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.155.5 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/cut pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

All of the *lj/cut* pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/cut/coul/long* and *lj/cut/tip4p/long* pair styles support the *pair_modify* table option since they can tabulate the short-range portion of the long-range Coulombic interaction.

All of the *lj/cut* pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure for the Lennard-Jones portion of the pair interaction.

All of the *lj/cut* pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The *lj/cut* and *lj/cut/coul/long* pair styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. The other styles only support the *pair* keyword of run_style respa. See the *run_style* command for details.

4.155.6 Restrictions

The *lj/cut/tip4p/long* styles are part of the KSPACE package. The *lj/cut/tip4p/cut* style is part of the MOLECULE package. These styles are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

4.155.7 Related commands

pair_coeff

4.155.8 Default

none

(Jorgensen) Jorgensen, Chandrasekhar, Madura, Impey, Klein, J Chem Phys, 79, 926 (1983).

4.156 pair_style lj/expand command

Accelerator Variants: *lj/expand/gpu*, *lj/expand/kk*, *lj/expand/omp*

4.157 pair_style lj/expand/coul/long command

Accelerator Variants: *lj/expand/coul/long/gpu*, *lj/expand/coul/long/kk*

4.157.1 Syntax

```
pair_style lj/expand cutoff
```

- cutoff = global cutoff for lj/expand interactions (distance units)

4.157.2 Examples

```
pair_style lj/expand 2.5
pair_coeff * * 1.0 1.0 0.5
pair_coeff 1 1 1.0 1.0 -0.2 2.0

pair_style lj/expand/coul/long 2.5
pair_style lj/expand/coul/long 2.5 4.0
pair_coeff * * 1.0 1.0 0.5
pair_coeff 1 1 1.0 1.0 -0.2 3.0
```

4.157.3 Description

Style *lj/expand* computes a LJ interaction with a distance shifted by delta which can be useful when particles are of different sizes, since it is different that using different sigma values in a standard LJ formula:

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r - \Delta} \right)^{12} - \left(\frac{\sigma}{r - \Delta} \right)^{6} \right] \qquad r < r_c + \Delta$$

r_c is the cutoff which does not include the Δ distance. I.e. the actual force cutoff is the sum of $r_c + \Delta$.

For all of the *lj/expand* pair styles, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- Δ (distance units)
- cutoff (distance units)

The Δ values can be positive or negative. The last coefficient is optional. If not specified, the global LJ cutoff is used.

For *lj/expand/coul/long* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.157.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon, sigma, and shift coefficients and cutoff distance for this pair style can be mixed. Shift is always mixed via an *arithmetic* rule. The other coefficients are mixed according to the pair_modify mix value. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style supports the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure of the pair interaction.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.157.5 Restrictions

none

4.157.6 Related commands

pair_coeff

4.157.7 Default

none

4.158 pair_style lj/expand/sphere command

Accelerator Variant: *lj/expand/sphere/omp*

4.158.1 Syntax

```
pair_style style args
```

- style = *lj/expand/sphere*
- args = list of arguments for a particular style

lj/expand/sphere args = cutoff
  cutoff = global cutoff for Lennard Jones interactions (distance units)

4.158.2 Examples

```
pair_style lj/expand/sphere 2.5
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.1 0.4 2.8
```

4.158.3 Description

Added in version 15Jun2023.

The *lj/expand/sphere* style compute a 12/6 Lennard-Jones potential with a distance shifted by $\Delta = \frac{1}{2}(d_i+d_j)$, the average diameter of both atoms. This can be used to model particles of different sizes but same interactions, which is different from using different sigma values as in *pair style lj/cut/sphere*.

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r-\Delta} \right)^{12} - \left(\frac{\sigma}{r-\Delta} \right)^{6} \right] \qquad r < r_c + \Delta$$

r_c is the cutoff which does not include the distance Δ. I.e. the actual force cutoff is the sum $r_c + \Delta$.

This is the same potential function used by the *lj/expand* pair style, but the Δ parameter is not set as a per-type parameter via the *pair_coeff command*. Instead it is calculated individually for each pair using the per-atom diameter attribute of *atom_style sphere* for the two atoms as the average diameter, $\Delta = \frac{1}{2}(d_i + d_j)$

Note that σ is defined in the LJ formula above as the zero-crossing distance for the potential, *not* as the energy minimum which is at $2^{\frac{1}{6}}\sigma$.

Notes on cutoffs, neighbor lists, and efficiency

If your system is mildly polydisperse, meaning the ratio of the diameter of the largest particle to the smallest is less than 2.0, then the neighbor lists built by the code should be reasonably efficient. Which means they will not contain too many particle pairs that do not interact. However, if your system is highly polydisperse (ratio > 2.0), the neighbor list build and force computations may be inefficient. There are two ways to try and speed up the simulations.

The first is to assign atoms to different atom types so that atoms of each type are similar in size. E.g. if particle diameters range from 1 to 5 use 4 atom types, ensuring atoms of type 1 have diameters from 1.0-2.0, type 2 from

2.0-3.0, etc. This will reduce the number of non-interacting pairs in the neighbor lists and thus reduce the time spent on computing pairwise interactions.

The second is to use the *neighbor multi* command which enabled a different algorithm for building neighbor lists. This will also require that you assign multiple atom types according to diameters, but will in addition use a more efficient size-dependent strategy to construct the neighbor lists and thus reduce the time spent on building neighbor lists.

Here are example input script commands using the first option for a highly polydisperse system:

```
units           lj
atom_style      sphere
lattice         fcc 0.8442
region          box block 0 10 0 10 0 10
create_box      2 box
create_atoms    1 box

# create atoms with random diameters from bimodal distribution
variable switch atom random(0.0,1.0,345634)
variable diam atom (v_switch<0.75)*normal(0.2,0.04,325)+(v_switch>=0.7)*normal(0.6,0.2,453)
set group all diameter v_diam

# assign type 2 to atoms with diameter > 0.35
variable large atom (2.0*radius)>0.35
group large variable large
set group large type 2

pair_style      lj/expand/sphere 2.0
pair_coeff      * * 1.0 0.5

neighbor 0.3 bin
```

Using multiple atom types speeds up the calculation for this example by more than 30 percent, but using the multi-style neighbor list does not provide a speedup.

4.158.4 Coefficients

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- LJ cutoff (distance units) (optional)

The last coefficient is optional. If not specified, the global LJ cutoff specified in the *pair_style command* is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.158.5 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon, sigma, and cutoff coefficients for the *lj/expand/sphere* pair style can be mixed. The default mixing style is *geometric*. See the *pair_modify command* for details.

The *lj/expand/sphere* pair style supports the *pair_modify shift* option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/expand/sphere* pair style does *not* support the *pair_modify* tail option for adding a long-range tail corrections to the energy and pressure.

The *lj/expand/sphere* pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does *not* support the *inner*, *middle*, *outer* keywords.

4.158.6 Restrictions

The *lj/expand/sphere* pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

4.158.7 Related commands

- *pair_coeff*
- *pair_style lj/cut*
- *pair_style lj/cut/sphere*

4.158.8 Default

none

4.159 pair_style lj/long/coul/long command

Accelerator Variants: *lj/long/coul/long/intel*, *lj/long/coul/long/omp*, *lj/long/coul/long/opt*

4.160 pair_style lj/long/tip4p/long command

Accelerator Variants: *lj/long/tip4p/long/omp*

4.160.1 Syntax

```
pair_style style args
```

- style = *lj/long/coul/long* or *lj/long/tip4p/long*
- args = list of arguments for a particular style

lj/long/coul/long args = flag_lj flag_coul cutoff (cutoff2)
  flag_lj = long or cut or off
    long = use Kspace long-range summation for dispersion 1/r^6 term
    cut = use a cutoff on dispersion 1/r^6 term
    off = omit disperion 1/r^6 term entirely
  flag_coul = long or off
    long = use Kspace long-range summation for Coulombic 1/r term
    off = omit Coulombic term
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
lj/long/tip4p/long args = flag_lj flag_coul otype htype btype atype qdist cutoff (cutoff2)
  flag_lj = long or cut
    long = use Kspace long-range summation for dispersion 1/r^6 term
    cut = use a cutoff
  flag_coul = long or off
    long = use Kspace long-range summation for Coulombic 1/r term
    off = omit Coulombic term
  otype,htype = atom types (numeric or type label) for TIP4P O and H
  btype,atype = bond and angle types (numeric or type label) for TIP4P waters
  qdist = distance from O atom to massless charge (distance units)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.160.2 Examples

```
pair_style lj/long/coul/long cut off 2.5
pair_style lj/long/coul/long cut long 2.5 4.0
pair_style lj/long/coul/long long long 2.5 4.0
pair_coeff * * 1 1
pair_coeff 1 1 1 3 4

pair_style lj/long/tip4p/long long long 1 2 7 8 0.15 12.0
pair_style lj/long/tip4p/long long long 1 2 7 8 0.15 12.0 10.0
pair_coeff * * 100.0 3.0
pair_coeff 1 1 100.0 3.5 9.0

pair_style lj/long/tip4p/long long long OW HW HW-OW HW-OW-HW 0.15 12.0
labelmap atom 1 OW 2 HW
labelmap bond 1 HW-OW
```

(continues on next page)

(continued from previous page)

```
labelmap angle 1 HW-OW-HW
pair_coeff * * 100.0 3.0
pair_coeff OW OW 100.0 3.5 9.0
```

4.160.3 Description

Style *lj/long/coul/long* computes the standard 12/6 Lennard-Jones potential:

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_c$$

with ε and σ being the usual Lennard-Jones potential parameters, plus the Coulomb potential, given by:

$$E = \frac{Cq_iq_j}{\varepsilon r} \qquad r < r_c$$

where C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, ε is the dielectric constant which can be set by the *dielectric* command, and r_c is the cutoff. If one cutoff is specified in the pair_style command, it is used for both the LJ and Coulombic terms. If two cutoffs are specified, they are used as cutoffs for the LJ and Coulombic terms respectively.

The purpose of this pair style is to capture long-range interactions resulting from both attractive 1/r^6 Lennard-Jones and Coulombic 1/r interactions. This is done by use of the *flag_lj* and *flag_coul* settings. The *In 't Veld* paper has more details on when it is appropriate to include long-range 1/r^6 interactions, using this potential.

Style *lj/long/tip4p/long* implements the TIP4P water model of *(Jorgensen)*, which introduces a massless site located a short distance away from the oxygen atom along the bisector of the HOH angle. The atomic types of the oxygen and hydrogen atoms, the bond and angle types for OH and HOH interactions, and the distance to the massless charge site are specified as pair_style arguments.

Note

For each TIP4P water molecule in your system, the atom IDs for the O and 2 H atoms must be consecutive, with the O atom first. This is to enable LAMMPS to "find" the 2 H atoms associated with each O atom. For example, if the atom ID of an O atom in a TIP4P water molecule is 500, then its 2 H atoms must have IDs 501 and 502.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

See the *Howto tip4p* page for more information on how to use the TIP4P pair style. Note that the neighbor list cutoff for Coulomb interactions is effectively extended by a distance 2*qdist when using the TIP4P pair style, to account for the offset distance of the fictitious charges on O atoms in water molecules. Thus it is typically best in an efficiency sense to use a LJ cutoff >= Coulombic cutoff + 2*qdist, to shrink the size of the neighbor list. This leads to slightly larger cost for the long-range calculation, so you can test the trade-off for your model.

If *flag_lj* is set to *long*, no cutoff is used on the LJ 1/r^6 dispersion term. The long-range portion can be calculated by using the *kspace_style ewald/disp or pppm/disp* commands. The specified LJ cutoff then determines which portion of the LJ interactions are computed directly by the pair potential versus which part is computed in reciprocal space via the Kspace style. If *flag_lj* is set to *cut*, the LJ interactions are simply cutoff, as with *pair_style lj/cut*.

If *flag_coul* is set to *long*, no cutoff is used on the Coulombic interactions. The long-range portion can calculated by using any of several *kspace_style* command options such as *pppm* or *ewald*. Note that if *flag_lj* is also set to long, then

the *ewald/disp* or *pppm/disp* Kspace style needs to be used to perform the long-range calculations for both the LJ and Coulombic interactions. If *flag_coul* is set to *off*, Coulombic interactions are not computed.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff1 (distance units)
- cutoff2 (distance units)

Note that sigma is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum at 2^(1/6) sigma.

The latter 2 coefficients are optional. If not specified, the global LJ and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both LJ and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the LJ and Coulombic cutoffs for this type pair.

Note that if you are using *flag_lj* set to *long*, you cannot specify a LJ cutoff for an atom type pair, since only one global LJ cutoff is allowed. Similarly, if you are using *flag_coul* set to *long*, you cannot specify a Coulombic cutoff for an atom type pair, since only one global Coulombic cutoff is allowed.

For *lj/long/tip4p/long* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

A version of these styles with a soft core, *lj/cut/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles. The version with soft core is only available if LAMMPS was built with that package. See the *Build package* page for more info.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.160.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/long pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

These pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction, assuming *flag_lj* is *cut*.

These pair styles support the *pair_modify* table and table/disp options since they can tabulate the short-range portion of the long-range Coulombic and dispersion interactions.

Thes pair styles do not support the *pair_modify* tail option for adding a long-range tail correction to the Lennard-Jones portion of the energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The pair lj/long/coul/long styles support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. See the *run_style* command for details.

4.160.5 Restrictions

These styles are part of the KSPACE package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.160.6 Related commands

pair_coeff

4.160.7 Default

none

(In 't Veld) In 't Veld, Ismail, Grest, J Chem Phys, 127, 144711 (2007).

(Jorgensen) Jorgensen, Chandrasekhar, Madura, Impey, Klein, J Chem Phys, 79, 926 (1983).

4.161 pair_style lj/relres command

Accelerator Variants: *lj/relres/omp*

4.161.1 Syntax

```
pair_style lj/relres Rsi Rso Rci Rco
```

- Rsi = inner switching cutoff between the fine-grained and coarse-grained potentials (distance units)
- Rso = outer switching cutoff between the fine-grained and coarse-grained potentials (distance units)
- Rci = inner cutoff beyond which the force smoothing for all interactions is applied (distance units)
- Rco = outer cutoff for all interactions (distance units)

4.161.2 Examples

```
pair_style lj/relres 4.0 5.0 8.0 10.0
pair_coeff 1 1 0.5 1.0 1.5 1.1
pair_coeff 2 2 0.5 1.0 0.0 0.0 3.0 3.5 6.0 7.0
```

4.161.3 Description

Pair style *lj/relres* computes a LJ interaction using the Relative Resolution (RelRes) framework which applies a fine-grained (FG) potential between near neighbors and a coarse-grained (CG) potential between far neighbors *(Chaimovich1)*. This approach can improve the computational efficiency by almost an order of magnitude, while maintaining the correct static and dynamic behavior of a reference system *(Chaimovich2)*.

$$E = \begin{cases} 4\varepsilon^{FG}\left[\left(\frac{\sigma^{FG}}{r}\right)^{12} - \left(\frac{\sigma^{FG}}{r}\right)^{6}\right] - \Gamma_{si}, & \text{if} \quad r < r_{si}, \\ \sum_{m=0}^{4} \gamma_{sm}\left(r - r_{si}\right)^{m} - \Gamma_{so}, & \text{if} \quad r_{si} \leq r < r_{so}, \\ 4\varepsilon^{CG}\left[\left(\frac{\sigma^{CG}}{r}\right)^{12} - \left(\frac{\sigma^{CG}}{r}\right)^{6}\right] - \Gamma_{c}, & \text{if} \quad r_{so} \leq r < r_{ci}, \\ \sum_{m=0}^{4} \gamma_{cm}\left(r - r_{ci}\right)^{m} - \Gamma_{c}, & \text{if} \quad r_{ci} \leq r < r_{co}, \\ 0, & \text{if} \quad r \geq r_{co}. \end{cases}$$

The FG parameters of the LJ potential (ε^{FG} and σ^{FG}) are applied up to the inner switching cutoff, r_{si}, while the CG parameters of the LJ potential (ε^{CG} and σ^{CG}) are applied beyond the outer switching cutoff, r_{so}. Between r_{si} and r_{so} a polynomial smoothing function is applied so that the force and its derivative are continuous between the FG and CG potentials. An analogous smoothing function is applied between the inner and outer cutoffs (r_{ci} and r_{co}). The offsets Γ_{si}, Γ_{so} and Γ_{c} ensure the continuity of the energy over the entire domain. The corresponding polynomial coefficients γ_{sm} and γ_{cm}, as well as the offsets are automatically computed by LAMMPS.

Note

Energy and force resulting from this methodology can be plotted via the *pair_write* command.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as will be described below:

- ε^{FG} (energy units)
- σ^{FG} (distance units)
- ε^{CG} (energy units)
- σ^{CG} (distance units)

Additional parameters can be defined to specify different r_{si}, r_{so}, r_{ci}, r_{co} for a particular set of atom types:

- r_{si} (distance units)
- r_{so} (distance units)
- r_{ci} (distance units)
- r_{co} (distance units)

These parameters are optional, and they are used to override the global cutoffs as defined in the pair_style command. If not specified, the global values for r_{si}, r_{so}, r_{ci}, and r_{co} are used. If this override option is employed, all four arguments must be specified.

Here are some guidelines for using the pair_style *lj/relres* command.

In general, RelRes focuses on the speedup of pairwise interactions between all LJ sites. Importantly, it works with any settings and flags (e.g., *special_bonds* settings and *newton* flags) that can be used in a molecular simulation with the conventional LJ potential. In particular, all intramolecular topology with its energetics (i.e., bonds, angles, etc.) remains unaltered.

At the most basic level in the RelRes framework, all sites are mapped into clusters. Each cluster is just a collection of sites bonded together (the bonds themselves are not part of the cluster). In general, a molecule may be comprised of several clusters, and preferably, no two sites in a cluster are separated by more than two bonds. There are two categories of sites in RelRes: "hybrid" sites embody both FG and CG models, while "ordinary" sites embody just FG characteristics with no CG features. A given cluster has a single hybrid site (typically its central site) and several ordinary sites (typically its peripheral sites). Notice that while clusters are necessary for the RelRes parameterization (discussed below), they are not actually defined in LAMMPS. Besides, the total number of sites in the cluster are called the "mapping ratio", and this substantially impacts the computational efficiency of RelRes: For a mapping ratio of 3, the efficiency factor is around 4, and for a mapping ratio of 5, the efficiency factor is around 5 *(Chaimovich2)*.

The flexibility of LAMMPS allows placing any values for the LJ parameters in the input script. However, here are the optimal recommendations for the RelRes parameters, which yield the correct structural and thermal behavior in a system of interest *(Chaimovich1)*. One must first assign a complete set of parameters for the FG interactions that are applicable to all atom types. Regarding the parameters for the CG interactions, the rules rely on the site category (if it is a hybrid or an ordinary site). For atom types of ordinary sites, ε^{CG} must be set to 0 (zero) while the specific value of σ^{CG} is irrelevant. For atom types of hybrid sites, the CG parameters should be generally calculated using the following equations:

$$\sigma_I^{CG} = \frac{\left(\left(\sum_{\alpha\in A}\sqrt{\varepsilon_\alpha^{FG}\left(\sigma_\alpha^{FG}\right)^{12}}\right)^{1/2}\right)}{\left(\left(\sum_{\alpha\in A}\sqrt{\varepsilon_\alpha^{FG}\left(\sigma_\alpha^{FG}\right)^{6}}\right)^{1/3}\right)} \quad \text{and} \quad \varepsilon_I^{CG} = \frac{\left(\left(\sum_{\alpha\in A}\sqrt{\varepsilon_\alpha^{FG}\left(\sigma_\alpha^{FG}\right)^{6}}\right)^{4}\right)}{\left(\left(\sum_{\alpha\in A}\sqrt{\varepsilon_\alpha^{FG}\left(\sigma_\alpha^{FG}\right)^{12}}\right)^{2}\right)}$$

where I is an atom type of a hybrid site of a particular cluster A, and corresponding with this cluster, the summation proceeds over all of its sites α. These equations are derived from the monopole term in the underlying Taylor series, and they are indeed relevant only if geometric mixing is applicable for the FG model; if this is not the case, Ref. *(Chaimovich2)* discusses the alternative formula, and in such a situation, the pair_coeff command should be explicitly used for all combinations of atom types I ! = J.

The switching distance (the midpoint between inner and outer switching cutoffs) is another crucial factor in RelRes: decreasing it improves the computational efficiency, yet if it is too small, the molecular simulations may not capture the system behavior correctly. As a rule of thumb, the switching distance should be approximately $\sim 1.5\sigma$ *(Chaimovich1)*; recommendations can be found in Ref. *(Chaimovich2)*. Regarding the switching smoothing zone, $\sim 0.1\sigma$ is recommended; if desired, smoothing can be eliminated by setting the inner switching cutoff, r_{si}, equal to the outer switching cutoff, r_{so} (the same is true for the other cutoffs r_{ci} and r_{co}).

As an example, imagine that in your system, a molecule is comprised just of one cluster such that one atom type (#1) is associated with its hybrid site, and another atom type (#2) is associated with its ordinary sites (in total, there are 2 atom types). If geometric mixing is applicable, the following commands should be used:

```
pair_style lj/relres Rsi Rso Rci Rco
pair_coeff 1 1 epsilon_FG1 sigma_FG1 epsilon_CG1 sigma_CG1
pair_coeff 2 2 epsilon_FG2 sigma_FG2 0.0         0.0
pair_modify shift yes
```

In a more complex situation, there may be two distinct clusters in a system (these two clusters may be on same molecule or on different molecules), each with its own switching cutoffs. If there are still two atom types in each cluster as in the earlier example, the commands should be:

```
pair_style lj/relres Rsi Rso Rci Rco
pair_coeff 1 1 epsilon_FG1 sigma_FG1 epsilon_CG1 sigma_CG1 Rsi1 Rso1 Rci Rco
pair_coeff 2 2 epsilon_FG2 sigma_FG2 0.0         0.0       Rsi1 Rso1 Rci Rco
pair_coeff 3 3 epsilon_FG3 sigma_FG3 epsilon_CG3 sigma_CG3
pair_coeff 4 4 epsilon_FG4 sigma_FG4 0.0         0.0
pair_modify shift yes
```

In this example, the switching cutoffs for the first cluster (atom types 1 and 2) is defined explicitly in the pair_coeff command which overrides the global values, while the second cluster (atom types 3 and 4) uses the global definition from the pair_style command. The emphasis here is that the atom types that belong to a specific cluster should have the same switching/cutoff arguments.

In the case that geometric mixing is not applicable, for simulating the system from the previous example, we recommend using the following commands:

```
pair_style lj/relres Rsi Rso Rci Rco
pair_coeff 1 1 epsilon_FG1  sigma_FG1  epsilon_CG1  sigma_CG1  Rsi1  Rso1  Rci Rco
pair_coeff 1 2 epsilon_FG12 sigma_FG12 0.0          0.0        Rsi1  Rso1  Rci Rco
pair_coeff 1 3 epsilon_FG13 sigma_FG13 epsilon_CG13 sigma_CG13 Rsi13 Rso13 Rci Rco
pair_coeff 1 4 epsilon_FG14 sigma_FG14 0.0          0.0        Rsi13 Rso13 Rci Rco
pair_coeff 2 2 epsilon_FG2  sigma_FG2  0.0          0.0        Rsi1  Rso1  Rci Rco
pair_coeff 2 3 epsilon_FG23 sigma_FG23 0.0          0.0        Rsi13 Rso13 Rci Rco
pair_coeff 2 4 epsilon_FG24 sigma_FG24 0.0          0.0        Rsi13 Rso13 Rci Rco
pair_coeff 3 3 epsilon_FG3  sigma_FG3  epsilon_CG3  sigma_CG3
pair_coeff 3 4 epsilon_FG34 sigma_FG34 0.0          0.0
pair_coeff 4 4 epsilon_FG4  sigma_FG4  0.0          0.0
pair_modify shift yes
```

Notice that the CG parameters are mixed only for interactions between atom types associated with hybrid sites, and that the cutoffs are mixed on the cluster basis.

More examples can be found in the *examples/relres* folder.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.161.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I, J with I ! = J, the ε^{FG}, σ^{FG}, ε^{CG}, σ^{CG}, r_{si}, r_{so}, r_{ci}, and r_{co} parameters for this pair style can be mixed, if not defined explicitly. All parameters are mixed according to the pair_modify mix option. The default mix value is *geometric*, and it is recommended to use with this *lj/relres* style. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction. It is recommended to set this option to *yes*. Otherwise, the offset Γ_c is set to zero. Constants Γ_{si} and Γ_{so} are not impacted by this option.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since the energy of the pair interaction is smoothed to 0.0 at the cutoff.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.161.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.161.6 Related commands

pair_coeff

4.161.7 Default

none

(Chaimovich1) A. Chaimovich, C. Peter and K. Kremer, J. Chem. Phys. 143, 243107 (2015).

(Chaimovich2) M. Chaimovich and A. Chaimovich, J. Chem. Theory Comput. 17, 1045-1059 (2021).

4.162 pair_style lj/smooth command

Accelerator Variants: *lj/smooth/gpu*, *lj/smooth/omp*

4.162.1 Syntax

```
pair_style lj/smooth Rin Rc
```

- Rin = inner cutoff beyond which force smoothing will be applied (distance units)
- Rc = outer cutoff for lj/smooth interactions (distance units)

4.162.2 Examples

```
pair_style lj/smooth 8.0 10.0
pair_coeff * * 10.0 1.5
pair_coeff 1 1 20.0 1.3 7.0 9.0
```

4.162.3 Description

Style *lj/smooth* computes a LJ interaction with a force smoothing applied between the inner and outer cutoff.

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \qquad r < r_{in}$$

$$F = C_1 + C_2(r - r_{in}) + C_3(r - r_{in})^2 + C_4(r - r_{in})^3 \qquad r_{in} < r < r_c$$

The polynomial coefficients C1, C2, C3, C4 are computed by LAMMPS to cause the force to vary smoothly from the inner cutoff r_{in} to the outer cutoff r_c.

At the inner cutoff the force and its first derivative will match the non-smoothed LJ formula. At the outer cutoff the force and its first derivative will be 0.0. The inner cutoff cannot be 0.0.

Note

this force smoothing causes the energy to be discontinuous both in its values and first derivative. This can lead to poor energy conservation and may require the use of a thermostat. Plot the energy and force resulting from this formula via the *pair_write* command to see the effect.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- r_{in} (distance units)
- r_c (distance units)

The last 2 coefficients are optional inner and outer cutoffs. If not specified, the global values for r_{in} and r_c are used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.162.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon, sigma, Rin coefficients and the cutoff distance for this pair style can be mixed. Rin is a cutoff value and is mixed like the cutoff. The other coefficients are mixed according to the pair_modify mix option. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since the energy of the pair interaction is smoothed to 0.0 at the cutoff.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.162.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.162.6 Related commands

pair_coeff , *pair lj/smooth/linear*

4.162.7 Default

none

4.163 pair_style lj/smooth/linear command

Accelerator Variants: *lj/smooth/linear/omp*

4.163.1 Syntax

```
pair_style lj/smooth/linear cutoff
```

- cutoff = global cutoff for Lennard-Jones interactions (distance units)

4.163.2 Examples

```
pair_style lj/smooth/linear 2.5
pair_coeff * * 1.0 1.0
pair_coeff 1 1 0.3 3.0 9.0
```

4.163.3 Description

Style *lj/smooth/linear* computes a truncated and force-shifted LJ interaction (aka Shifted Force Lennard-Jones) that combines the standard 12/6 Lennard-Jones function and subtracts a linear term based on the cutoff distance, so that both, the potential and the force, go continuously to zero at the cutoff r_c *(Toxvaerd)*:

$$\phi(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

$$E(r) = \phi(r) - \phi(r_c) - (r - r_c)\left.\frac{d\phi}{dr}\right|_{r=r_c} \qquad r < r_c$$

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global LJ cutoff specified in the pair_style command is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.163.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance can be mixed. The default mix value is geometric. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction, since it goes to 0.0 at the cutoff by construction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since the energy of the pair interaction is smoothed to 0.0 at the cutoff.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.163.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.163.6 Related commands

pair_coeff, *pair lj/smooth*

4.163.7 Default

none

(Toxvaerd) Toxvaerd, Dyre, J Chem Phys, 134, 081102 (2011).

4.164 pair_style lj/switch3/coulgauss/long command

4.165 pair_style mm3/switch3/coulgauss/long command

4.165.1 Syntax

```
pair_style style args
```

- style = *lj/switch3/coulgauss/long* or *mm3/switch3/coulgauss/long*
- args = list of arguments for a particular style

lj/switch3/coulgauss/long args = cutoff (cutoff2) width
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
  width = width parameter of the smoothing function (distance units)

mm3/switch3/coulgauss/long args = cutoff (cutoff2) width
  cutoff = global cutoff for MM3 (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)
  width = width parameter of the smoothing function (distance units)

4.165.2 Examples

```
pair_style lj/switch3/coulgauss/long    12.0 3.0
pair_coeff 1  0.2 2.5 1.2

pair_style lj/switch3/coulgauss/long   12.0 10.0 3.0
pair_coeff 1  0.2 2.5 1.2

pair_style mm3/switch3/coulgauss/long    12.0 3.0
pair_coeff 1  0.2 2.5 1.2

pair_style mm3/switch3/coulgauss/long   12.0 10.0 3.0
pair_coeff 1  0.2 2.5 1.2
```

4.165.3 Description

The *lj/switch3/coulgauss* style evaluates the LJ vdW potential

$$E = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

The *mm3/switch3/coulgauss/long* style evaluates the MM3 vdW potential *(Allinger)*

$$E = \varepsilon_{ij} \left[-2.25 \left(\frac{r_{v,ij}}{r_{ij}}\right)^{6} + 1.84(10)^5 \exp\left[-12.0 r_{ij}/r_{v,ij}\right] \right] S_3(r_{ij})$$

$$r_{v,ij} = r_{v,i} + r_{v,j}$$

$$\varepsilon_{ij} = \sqrt{\varepsilon_i \varepsilon_j}$$

Both potentials go smoothly to zero at the cutoff r_c as defined by the switching function

$$S_3(r) = \begin{cases} 1 & \text{if} \quad r < r_c - w \\ 3x^2 - 2x^3 & \text{if} \quad r < r_c \quad \text{with} \quad x = \frac{r_c - r}{w} \\ 0 & \text{if} \quad r >= r_c \end{cases}$$

where w is the width defined in the arguments. This potential is combined with Coulomb interaction between Gaussian charge densities:

$$E = \frac{q_i q_j \mathrm{erf}\left(r/\sqrt{\gamma_1^2 + \gamma_2^2}\right)}{\varepsilon r_{ij}}$$

where q_i and q_j are the charges on the two atoms, ε is the dielectric constant which can be set by the *dielectric* command, γ_i and γ_j are the widths of the Gaussian charge distribution and erf() is the error-function. This style has to be used in conjunction with the *kspace_style* command

If one cutoff is specified it is used for both the vdW and Coulomb terms. If two cutoffs are specified, the first is used as the cutoff for the vdW terms, and the second is the cutoff for the Coulombic term.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- ε (energy)
- σ (distance)
- γ (distance)

4.165.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/long pair styles can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

Shifting the potential energy is not necessary because the switching function ensures that the potential is zero at the cut-off.

These pair styles support the *pair_modify* table and options since they can tabulate the short-range portion of the long-range Coulombic interactions.

Thes pair styles do not support the *pair_modify* tail option for adding a long-range tail correction to the Lennard-Jones portion of the energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.165.5 Restrictions

These styles are part of the YAFF package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.165.6 Related commands

pair_coeff

4.165.7 Default

none

4.166 pair_style local/density command

4.166.1 Syntax

```
pair_style style arg
```

- style = *local/density*
- arg = name of file containing tabulated values of local density and the potential

4.166.2 Examples

```
pair_style local/density benzene_water.localdensity.table

pair_style hybrid/overlay table spline 500 local/density
pair_coeff * * local/density  benzene_water.localdensity.table
```

4.166.3 Description

The local density (LD) potential is a mean-field manybody potential, and, in some way, a generalization of embedded atom models (EAM). The name "local density potential" arises from the fact that it assigns an energy to an atom depending on the number of neighboring atoms of a given type around it within a predefined spherical volume (i.e., within the cutoff). The bottom-up coarse-graining (CG) literature suggests that such potentials can be widely useful in capturing effective multibody forces in a computationally efficient manner and thus improve the quality of CG models of implicit solvation *(Sanyal1)* and phase-segregation in liquid mixtures *(Sanyal2)*, and provide guidelines to determine the extent of manybody correlations present in a CG model *(Rosenberger)*. The LD potential in LAMMPS is primarily intended to be used as a corrective potential over traditional pair potentials in bottom-up CG models via *hybrid/overlay pair style* with other explicit pair interaction terms (e.g., tabulated, Lennard-Jones, Morse etc.). Because the LD potential is not a pair potential per se, it is implemented simply as a single auxiliary file with all specifications that will be read upon initialization.

Note

Thus when used as the only interaction in the system, there is no corresponding pair_coeff command and when used with other pair styles using the hybrid/overlay option, the corresponding pair_coeff command must be supplied * * as placeholders for the atom types.

System with a single CG atom type:

A system of a single atom type (e.g., LJ argon) with a single local density (LD) potential would have an energy given by:

$$U_{LD} = \sum_i F(\rho_i)$$

where ρ_i is the LD at atom i and $F(\rho)$ is similar in spirit to the embedding function used in EAM potentials. The LD at atom i is given by the sum

$$\rho_i = \sum_{j \neq i} \varphi(r_{ij})$$

where φ is an indicator function that is one at r=0 and zero beyond a cutoff distance R2. The choice of the functional form of φ is somewhat arbitrary, but the following piecewise cubic function has proven sufficiently general: *(Sanyal1)*, *(Sanyal2)* *(Rosenberger)*

$$\varphi(r) = \begin{cases} 1 & r \leq R_1 \\ c_0 + c_2 r^2 + c_4 r^4 + c_6 r^6 & r \in (R_1, R_2) \\ 0 & r \geq R_2 \end{cases}$$

The constants c are chosen so that the indicator function smoothly interpolates between 1 and 0 between the distances R1 and R2, which are called the inner and outer cutoffs, respectively. Thus phi satisfies phi(R1) = 1, phi(R2) = dphi/dr @ (r=R1) = dphi/dr @ (r=R2) = 0. The embedding function F(rho) may or may not have a closed-form expression. To maintain generality, it is practically represented with a spline-interpolated table over a predetermined range of rho. Outside of that range it simply adopts zero values at the endpoints.

It can be shown that the total force between two atoms due to the LD potential takes the form of a pair force, which motivates its designation as a LAMMPS pair style. Please see *(Sanyal1)* for details of the derivation.

Systems with arbitrary numbers of atom types:

The potential is easily generalized to systems involving multiple atom types:

$$U_{LD} = \sum_i a_\alpha F(\rho_i)$$

with the LD expressed as

$$\rho_i = \sum_{j \neq i} b_\beta \varphi(r_{ij})$$

where α gives the type of atom i, β the type of atom j, and the coefficients a and b filter for atom types as specified by the user. a is called the central atom filter as it determines to which atoms the potential applies; $a_\alpha = 1$ if the LD potential applies to atom type α else zero. On the other hand, b is called the neighbor atom filter because it specifies which atom types to use in the calculation of the LD; $b_\beta = 1$ if atom type β contributes to the LD and zero otherwise.

Note

Note that the potentials need not be symmetric with respect to atom types, which is the reason for two distinct sets of coefficients a and b. An atom type may contribute to the LD but not the potential, or to the potential but not the LD. Such decisions are made by the user and should (ideally) be motivated on physical grounds for the problem at hand.

General form for implementation in LAMMPS:

Of course, a system with many atom types may have many different possible LD potentials, each with their own atom type filters, cutoffs, and embedding functions. The most general form of this potential as implemented in the pair_style local/density is:

$$U_{LD} = \sum_k U_{LD}^{(k)} = \sum_i \left[\sum_k a_\alpha^{(k)} F^{(k)} \left(\rho_i^{(k)} \right) \right]$$

where, k is an index that spans the (arbitrary) number of applied LD potentials N_LD. Each LD is calculated as before with:

$$\rho_i^{(k)} = \sum_j b_\beta^{(k)} \varphi^{(k)}(r_{ij})$$

The superscript on the indicator function phi simply indicates that it is associated with specific values of the cutoff distances R1(k) and R2(k). In summary, there may be N_LD distinct LD potentials. With each potential type (k), one must specify:

- the inner and outer cutoffs as R1 and R2
- the central type filter a(k), where k = 1,2,...N_LD
- the neighbor type filter b(k), where k = 1,2,...N_LD
- the LD potential function F(k)(rho), typically as a table that is later spline-interpolated

---

Tabulated input file format:

```
Line 1:          comment or blank (ignored)
Line 2:          comment or blank (ignored)
Line 3:          N_LD N_rho (# of LD potentials and # of tabulated values, single space separated)
Line 4:          blank (ignored)
Line 5:          R1(k) R2(k) (lower and upper cutoffs, single space separated)
Line 6:          central-types (central atom types, single space separated)
Line 7:          neighbor-types (neighbor atom types single space separated)
Line 8:          rho_min rho_max drho (min, max and diff. in tabulated rho values, single space
↪separated)
Line 9:          F(k)(rho_min + 0.drho)
Line 10:         F(k)(rho_min + 1.drho)
Line 11:         F(k)(rho_min + 2.drho)
...
Line 9+N_rho:    F(k)(rho_min + N_rho . drho)
Line 10+N_rho:   blank (ignored)

Block 2

Block 3

Block N_LD
```

Lines 5 to 9+N_rho constitute the first block. Thus the input file is separated (by blank lines) into N_LD blocks each representing a separate LD potential and each specifying its own upper and lower cutoffs, central and neighbor atoms, and potential. In general, blank lines anywhere are ignored.

4.166.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support automatic mixing. For atom type pairs α, β and α != β, even if LD potentials of type (α, α) and (β, β) are provided, you will need to explicitly provide LD potential types (α, β) and (β, α) if need be (Here, the notation (α, β) means that α is the central atom to which the LD potential is applied and β is the neighbor atom which contributes to the LD potential on α).

This pair style does not support the *pair_modify* shift, table, and tail options.

The local/density pair style does not write its information to *binary restart files*, since it is stored in tabulated potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.166.5 Restrictions

The local/density pair style is a part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.166.6 Related commands

pair_coeff

4.166.7 Default

none

(Sanyal1) Sanyal and Shell, Journal of Chemical Physics, 2016, 145 (3), 034109.

(Sanyal2) Sanyal and Shell, Journal of Physical Chemistry B, 122 (21), 5678-5693.

(Rosenberger) Rosenberger, Sanyal, Shell and van der Vegt, Journal of Chemical Physics, 2019, 151 (4), 044111.

4.167 pair_style lubricate command

Accelerator Variants: *lubricate/omp*

4.168 pair_style lubricate/poly command

Accelerator Variants: *lubricate/poly/omp*

4.168.1 Syntax

```
pair_style style mu flaglog flagfld cutinner cutoff flagHI flagVF
```

- style = *lubricate* or *lubricate/poly*
- mu = dynamic viscosity (dynamic viscosity units)
- flaglog = 0/1 to exclude/include log terms in the lubrication approximation
- flagfld = 0/1 to exclude/include Fast Lubrication Dynamics (FLD) effects
- cutinner = inner cutoff distance (distance units)
- cutoff = outer cutoff for interactions (distance units)
- flagHI (optional) = 0/1 to exclude/include 1/r hydrodynamic interactions
- flagVF (optional) = 0/1 to exclude/include volume fraction corrections in the long-range isotropic terms

4.168.2 Examples

(all assume radius = 1)

```
pair_style lubricate 1.5 1 1 2.01 2.5
pair_coeff 1 1 2.05 2.8
pair_coeff * *

pair_style lubricate 1.5 1 1 2.01 2.5
pair_coeff * *
variable mu equal ramp(1,2)
fix 1 all adapt 1 pair lubricate mu * * v_mu
```

4.168.3 Description

Styles *lubricate* and *lubricate/poly* compute hydrodynamic interactions between mono-disperse finite-size spherical particles in a pairwise fashion. The interactions have 2 components. The first is Ball-Melrose lubrication terms via the formulas in *(Ball and Melrose)*

$$W = -a_{sq}|(v_1 - v_2) \bullet \mathbf{nn}|^2 - a_{sh}|(\omega_1 + \omega_2) \bullet (\mathbf{I} - \mathbf{nn}) - 2\Omega_N|^2 - a_{pu}|(\omega_1 - \omega_2) \bullet (\mathbf{I} - \mathbf{nn})|^2 - a_{tw}|(\omega_1 - \omega_2) \bullet \mathbf{nn}|^2 \qquad r < r_c$$

$$\Omega_N = \mathbf{n} \times (v_1 - v_2)/r$$

which represents the dissipation W between two nearby particles due to their relative velocities in the presence of a background solvent with viscosity *mu*. Note that this is dynamic viscosity which has units of mass/distance/time, not kinematic viscosity.

The Asq (squeeze) term is the strongest and is included if *flagHI* is set to 1 (default). It scales as 1/gap where gap is the separation between the surfaces of the 2 particles. The Ash (shear) and Apu (pump) terms are only included if *flaglog* is set to 1. They are the next strongest interactions, and the only other singular interaction, and scale as log(gap). Note that *flaglog* = 1 and *flagHI* = 0 is invalid, and will result in a warning message, after which *flagHI* will be set to 1. The Atw (twist) term is currently not included. It is typically a very small contribution to the lubrication forces.

The *flagHI* and *flagVF* settings are optional. Neither should be used, or both must be defined.

Cutinner sets the minimum center-to-center separation that will be used in calculations irrespective of the actual separation. *Cutoff* is the maximum center-to-center separation at which an interaction is computed. Using a *cutoff* less than 3 radii is recommended if *flaglog* is set to 1.

The other component is due to the Fast Lubrication Dynamics (FLD) approximation, described in *(Kumar)*, which can be represented by the following equation

$$F^H = -R_{FU}(U - U^\infty) + R_{FE}E^\infty$$

where U represents the velocities and angular velocities of the particles, U^∞ represents the velocity and the angular velocity of the undisturbed fluid, and E^∞ represents the rate of strain tensor of the undisturbed fluid with viscosity *mu*. Again, note that this is dynamic viscosity which has units of mass/distance/time, not kinematic viscosity. Volume fraction corrections to R_FU are included as long as *flagVF* is set to 1 (default).

Note

When using the FLD terms, these pair styles are designed to be used with explicit time integration and a correspondingly small timestep. Thus either *fix nve/sphere* or *fix nve/asphere* should be used for time integration. To perform implicit FLD, see the *pair_style lubricateU* command.

Style *lubricate* requires monodisperse spherical particles; style *lubricate/poly* allows for polydisperse spherical particles.

The viscosity *mu* can be varied in a time-dependent manner over the course of a simulation, in which case in which case the pair_style setting for *mu* will be overridden. See the *fix adapt* command for details.

If the suspension is sheared via the *fix deform* command then the pair style uses the shear rate to adjust the hydrodynamic interactions accordingly. Volume changes due to fix deform are accounted for when computing the volume fraction corrections to R_FU.

When computing the volume fraction corrections to R_FU, the presence of walls (whether moving or stationary) will affect the volume fraction available to colloidal particles. This is currently accounted for with the following types of walls: *wall/lj93*, *wall/lj126*, *wall/colloid*, and *wall/harmonic*. For these wall styles, the correct volume fraction will be used when walls do not coincide with the box boundary, as well as when walls move and thereby cause a change in the volume fraction. Other wall styles may still work, but they will result in the volume fraction being computed based on the box boundaries. Several wall styles are not compatible with these pair styles and using them will result in an error.

Since lubrication forces are dissipative, it is usually desirable to thermostat the system at a constant temperature. If Brownian motion (at a constant temperature) is desired, it can be set using the *pair_style brownian* command. These pair styles and the brownian style should use consistent parameters for *mu*, *flaglog*, *flagfld*, *cutinner*, *cutoff*, *flagHI* and *flagVF*.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutinner (distance units)
- cutoff (distance units)

The two coefficients are optional. If neither is specified, the two cutoffs specified in the pair_style command are used. Otherwise both must be specified.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.168.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the two cutoff distances for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.168.5 Restrictions

These styles are part of the COLLOID package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Only spherical monodisperse particles are allowed for pair_style lubricate.

Only spherical particles are allowed for pair_style lubricate/poly.

These pair styles will not restart exactly when using the *read_restart* command, though they should provide statistically similar results. This is because the forces they compute depend on atom velocities. See the *read_restart* command for more details.

4.168.6 Related commands

pair_coeff, *pair_style lubricateU*

4.168.7 Default

The default settings for the optional args are flagHI = 1 and flagVF = 1.

(Ball) Ball and Melrose, Physica A, 247, 444-472 (1997).

(Kumar) Kumar and Higdon, Phys Rev E, 82, 051401 (2010). See also his thesis for more details: A. Kumar, "Microscale Dynamics in Suspensions of Non-spherical Particles", Thesis, University of Illinois Urbana-Champaign, (2010). (https://www.ideals.illinois.edu/handle/2142/16032)

4.169 pair_style lubricateU command

4.170 pair_style lubricateU/poly command

4.170.1 Syntax

```
pair_style style mu flaglog cutinner cutoff gdot flagHI flagVF
```

- style = *lubricateU* or *lubricateU/poly*
- mu = dynamic viscosity (dynamic viscosity units)
- flaglog = 0/1 to exclude/include log terms in the lubrication approximation
- cutinner = inner cut off distance (distance units)
- cutoff = outer cutoff for interactions (distance units)
- gdot = shear rate (1/time units)
- flagHI (optional) = 0/1 to exclude/include 1/r hydrodynamic interactions
- flagVF (optional) = 0/1 to exclude/include volume fraction corrections in the long-range isotropic terms

4.170.2 Examples

(all assume radius = 1)

```
pair_style lubricateU 1.5 1 2.01 2.5 0.01 1 1
pair_coeff 1 1 2.05 2.8
pair_coeff * *
```

4.170.3 Description

Styles *lubricateU* and *lubricateU/poly* compute velocities and angular velocities for finite-size spherical particles such that the hydrodynamic interaction balances the force and torque due to all other types of interactions.

The interactions have 2 components. The first is Ball-Melrose lubrication terms via the formulas in *(Ball and Melrose)*

$$W = -a_{sq}|(v_1 - v_2) \bullet \mathbf{nn}|^2 - a_{sh}|(\omega_1 + \omega_2) \bullet (\mathbf{I} - \mathbf{nn}) - 2\Omega_N|^2 - a_{pu}|(\omega_1 - \omega_2) \bullet (\mathbf{I} - \mathbf{nn})|^2 - a_{tw}|(\omega_1 - \omega_2) \bullet \mathbf{nn}|^2 \qquad r < r_c$$

$$\Omega_N = \mathbf{n} \times (v_1 - v_2)/r$$

which represents the dissipation W between two nearby particles due to their relative velocities in the presence of a background solvent with viscosity *mu*. Note that this is dynamic viscosity which has units of mass/distance/time, not kinematic viscosity.

The Asq (squeeze) term is the strongest and is included as long as *flagHI* is set to 1 (default). It scales as 1/gap where gap is the separation between the surfaces of the 2 particles. The Ash (shear) and Apu (pump) terms are only included if *flaglog* is set to 1. They are the next strongest interactions, and the only other singular interaction, and scale as log(gap). Note that *flaglog* = 1 and *flagHI* = 0 is invalid, and will result in a warning message, after which *flagHI* will be set to 1. The Atw (twist) term is currently not included. It is typically a very small contribution to the lubrication forces.

The *flagHI* and *flagVF* settings are optional. Neither should be used, or both must be defined.

Cutinner sets the minimum center-to-center separation that will be used in calculations irrespective of the actual separation. *Cutoff* is the maximum center-to-center separation at which an interaction is computed. Using a *cutoff* less than 3 radii is recommended if *flaglog* is set to 1.

The other component is due to the Fast Lubrication Dynamics (FLD) approximation, described in *(Kumar)*. The equation being solved to balance the forces and torques is

$$-R_{FU}(U - U^{\infty}) = -R_{FE}E^{\infty} - F^{rest}$$

where U represents the velocities and angular velocities of the particles, U^{∞} represents the velocities and the angular velocities of the undisturbed fluid, and E^{∞} represents the rate of strain tensor of the undisturbed fluid flow with viscosity *mu*. Again, note that this is dynamic viscosity which has units of mass/distance/time, not kinematic viscosity. Volume fraction corrections to R_FU are included if *flagVF* is set to 1 (default).

F*rest* represents the forces and torques due to all other types of interactions, e.g. Brownian, electrostatic etc. Note that this algorithm neglects the inertial terms, thereby removing the restriction of resolving the small interial time scale, which may not be of interest for colloidal particles. These pair styles solve for the velocity such that the hydrodynamic force balances all other types of forces, thereby resulting in a net zero force (zero inertia limit). When defining these pair styles, they must be defined last so that when these styles are invoked all other types of forces have already been computed. For the same reason, they won't work if additional non-pair styles are defined (such as bond or Kspace forces) as they are calculated in LAMMPS after the pairwise interactions have been computed.

Note

When using these styles, the pair styles are designed to be used with implicit time integration and a correspondingly larger timestep. Thus either *fix nve/noforce* should be used for spherical particles defined via *atom_style sphere* or *fix nve/asphere/noforce* should be used for spherical particles defined via *atom_style ellipsoid*. This is because the velocity and angular momentum of each particle is set by the pair style, and should not be reset by the time integration fix.

Style *lubricateU* requires monodisperse spherical particles; style *lubricateU/poly* allows for polydisperse spherical particles.

If the suspension is sheared via the *fix deform* command then the pair style uses the shear rate to adjust the hydrodynamic interactions accordingly. Volume changes due to fix deform are accounted for when computing the volume fraction corrections to R_FU.

When computing the volume fraction corrections to R_FU, the presence of walls (whether moving or stationary) will affect the volume fraction available to colloidal particles. This is currently accounted for with the following types of walls: *wall/lj93*, *wall/lj126*, *wall/colloid*, and *wall/harmonic*. For these wall styles, the correct volume fraction will be used when walls do not coincide with the box boundary, as well as when walls move and thereby cause a change in the volume fraction. To use these wall styles with pair_style *lubricateU* or *lubricateU/poly*, the *fld yes* option must be specified in the fix wall command. Other wall styles may still work, but they will result in the volume fraction being computed based on the box boundaries. Several wall styles are not compatible with these pair styles and using them will result in an error.

Since lubrication forces are dissipative, it is usually desirable to thermostat the system at a constant temperature. If Brownian motion (at a constant temperature) is desired, it can be set using the *pair_style brownian* command. These pair styles and the brownian style should use consistent parameters for *mu*, *flaglog*, *flagfld*, *cutinner*, *cutoff*, *flagHI* and *flagVF*.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutinner (distance units)
- cutoff (distance units)

The two coefficients are optional. If neither is specified, the two cutoffs specified in the pair_style command are used. Otherwise both must be specified.

4.170.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the two cutoff distances for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

These pair styles do not support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for these pair styles.

These pair styles do not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They does not support the *inner*, *middle*, *outer* keywords.

4.170.5 Restrictions

These styles are part of the COLLOID package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Currently, these pair styles assume that all other types of forces/torques on the particles have been already been computed when it is invoked. This requires this style to be defined as the last of the pair styles, and that no fixes apply additional constraint forces. One exception is the *fix wall/colloid* commands, which has an "fld" option to apply their wall forces correctly.

Only spherical monodisperse particles are allowed for pair_style lubricateU.

Only spherical particles are allowed for pair_style lubricateU/poly.

For sheared suspensions, it is assumed that the shearing is done in the xy plane, with x being the velocity direction and y being the velocity-gradient direction. In this case, one must use *fix deform* with the same rate of shear (erate).

These pair styles are only compatible with the following wall fixes: doc:*fix wall/lj93, fix wall/lj126, fix wall/lj1043, fix wall/colloid, fix wall/harmonic, fix wall/lepton, fix wall/morse, fix wall/table <fix_wall>*.

4.170.6 Related commands

pair_coeff , *pair_style lubricate*

4.170.7 Default

The default settings for the optional args are flagHI = 1 and flagVF = 1.

(Ball) Ball and Melrose, Physica A, 247, 444-472 (1997).

(Kumar) Kumar and Higdon, Phys Rev E, 82, 051401 (2010).

4.171 pair_style lj/mdf command

4.172 pair_style buck/mdf command

4.173 pair_style lennard/mdf command

4.173.1 Syntax

```
pair_style style args
```

- style = *lj/mdf* or *buck/mdf* or *lennard/mdf*
- args = list of arguments for a particular style

  lj/mdf args = cutoff1 cutoff2
    cutoff1 = inner cutoff for the start of the tapering function
    cutoff1 = out cutoff for the end of the tapering function
  buck/mdf args = cutoff1 cutoff2
    cutoff1 = inner cutoff for the start of the tapering function
    cutoff1 = out cutoff for the end of the tapering function
  lennard/mdf args = cutoff1 cutoff2
    cutoff1 = inner cutoff for the start of the tapering function
    cutoff1 = out cutoff for the end of the tapering function

4.173.2 Examples

```
pair_style lj/mdf 2.5 3.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.1 2.8 3.0 3.2

pair_style buck/mdf 2.5 3.0
pair_coeff * * 100.0 1.5 200.0
pair_coeff * * 100.0 1.5 200.0 3.0 3.5

pair_style lennard/mdf 2.5 3.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1021760.3664 2120.317338 3.0 3.2
```

4.173.3 Description

The *lj/mdf*, *buck/mdf* and *lennard/mdf* compute the standard 12-6 Lennard-Jones and Buckingham potential with the addition of a taper function that ramps the energy and force smoothly to zero between an inner and outer cutoff.

$$E_{smooth}(r) = E(r) * f(r)$$

The tapering, *f(r)*, is done by using the Mei, Davenport, Fernando function *(Mei)*.

$$\begin{aligned} f(r) &= 1.0 && \text{for} \quad r < r_m \\ f(r) &= (1-x)^3 * (1+3x+6x^2) && \text{for} \quad r_m < r < r_{cut} \\ f(r) &= 0.0 && \text{for} \quad r >= r_{cut} \end{aligned}$$

where

$$x = \frac{(r - r_m)}{(r_{cut} - r_m)}$$

Here r_m is the inner cutoff radius and r_{cut} is the outer cutoff radius.

For the *lj/mdf* pair_style, the potential energy, *E(r)*, is the standard 12-6 Lennard-Jones written in the epsilon/sigma form:

$$E(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

Either the first two or all of the following coefficients must be defined for each pair of atoms types via the pair_coeff command as in the examples above, or in the data file read by the *read_data*. The two cutoffs default to the global values and ε and σ can also be determined by mixing as described below:

- ε (energy units)
- σ (distance units)
- r_m (distance units)
- r_{cut} (distance units)

For the *buck/mdf* pair_style, the potential energy, *E(r)*, is the standard Buckingham potential with three required coefficients. The two cutoffs can be omitted and default to the corresponding global values:

$$E(r) = Ae^{(-r/\rho)} - \frac{C}{r^6}$$

- A (energy units)
- ρ (distance units)
- C (energy-distance^6 units)
- r_m (distance units)
- r_{cut} (distance units)

For the *lennard/mdf* pair_style, the potential energy, *E(r)*, is the standard 12-6 Lennard-Jones written in the A/B form:

$$E(r) = \frac{A}{r^{12}} - \frac{B}{r^6}$$

The following coefficients must be defined for each pair of atoms types via the pair_coeff command as in the examples above, or in the data file read by the read_data commands, or by mixing as described below. The two cutoffs default to their global values and must be either both given or both left out:

- A (energy-distance^12 units)
- B (energy-distance^6 units)
- r_m (distance units)
- r_{cut} (distance units)

4.173.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the ϵ and σ coefficients and cutoff distances for the lj/mdf pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details. The other two pair styles buck/mdf and lennard/mdf do not support mixing, so all I,J pairs of coefficients must be specified explicitly.

None of the lj/mdf, buck/mdf, or lennard/mdf pair styles supports the *pair_modify* shift option or long-range tail corrections to pressure and energy.

These styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.173.5 Restrictions

These pair styles can only be used if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* doc page for more info.

4.173.6 Related commands

pair_coeff

4.173.7 Default

none

(Mei) Mei, Davenport, Fernando, Phys Rev B, 43 4653 (1991)

4.174 pair_style meam command

Accelerator Variants: *meam/kk*

4.175 pair_style meam/ms command

Accelerator Variants: *meam/ms/kk*

4.175.1 Syntax

```
pair_style style
```

- style = *meam* or *meam/ms*

4.175.2 Examples

```
pair_style meam
pair_coeff * * ../potentials/library.meam Si ../potentials/si.meam Si
pair_coeff * * ../potentials/library.meam Ni Al NULL Ni Al Ni Ni

pair_style meam/ms
pair_coeff * * ../potentials/library.msmeam H Ga ../potentials/HGa.meam H Ga
```

4.175.3 Description

Note

The behavior of the MEAM potential for alloy systems has changed as of November 2010; see description below of the mixture_ref_t parameter

Pair style *meam* computes non-bonded interactions for a variety of materials using the modified embedded-atom method (MEAM) *(Baskes)*. Conceptually, it is an extension to the original *EAM method* which adds angular forces. It is thus suitable for modeling metals and alloys with fcc, bcc, hcp and diamond cubic structures, as well as materials with covalent interactions like silicon and carbon.

The *meam* pair style is a translation of the original Fortran version to C++. It is functionally equivalent but more efficient and has additional features. The Fortran version of the *meam* pair style has been removed from LAMMPS after the 12 December 2018 release.

Pair style *meam/ms* uses the multi-state MEAM (MS-MEAM) method according to *(Baskes2)*, which is an extension to MEAM. This pair style is mostly equivalent to *meam* and differs only where noted in the documentation below.

In the MEAM formulation, the total energy E of a system of atoms is given by:

$$E = \sum_i \left\{ F_i(\bar{\rho}_i) + \frac{1}{2} \sum_{i \neq j} \phi_{ij}(r_{ij}) \right\}$$

where F is the embedding energy which is a function of the atomic electron density ρ, and ϕ is a pair potential interaction. The pair interaction is summed over all neighbors J of atom I within the cutoff distance. As with EAM, the multi-body nature of the MEAM potential is a result of the embedding energy term. Details of the computation of the embedding and pair energies, as implemented in LAMMPS, are given in *(Gullet)* and references therein.

The various parameters in the MEAM formulas are listed in two files which are specified by the *pair_coeff* command. These are ASCII text files in a format consistent with other MD codes that implement MEAM potentials, such as the

serial DYNAMO code and Warp. Several MEAM potential files with parameters for different materials are included in the "potentials" directory of the LAMMPS distribution with a ".meam" suffix. All of these are parameterized in terms of LAMMPS *metal units*.

Note that unlike for other potentials, cutoffs for MEAM potentials are not set in the pair_style or pair_coeff command; they are specified in the MEAM potential files themselves.

Only a single pair_coeff command is used with the *meam* style which specifies two MEAM files and the element(s) to extract information for. The MEAM elements are mapped to LAMMPS atom types by specifying N additional arguments after the second filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- MEAM library file
- Element1, Element2, ...
- MEAM parameter file
- N element names = mapping of MEAM elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential files.

As an example, the potentials/library.meam file has generic MEAM settings for a variety of elements. The potentials/SiC.meam file has specific parameter settings for a Si and C alloy system. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * library.meam Si C sic.meam Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first filename is the element library file. The list of elements following it extracts lines from the library file and assigns numeric indices to these elements. The second filename is the alloy parameter file, which refers to elements using the numeric indices assigned before. The arguments after the parameter file map LAMMPS atom types to elements, i.e. LAMMPS atom types 1,2,3 to the MEAM Si element. The final C argument maps LAMMPS atom type 4 to the MEAM C element.

If the second filename is specified as NULL, no parameter file is read, which simply means the generic parameters in the library file are used. Use of the NULL specification for the parameter file is discouraged for systems with more than a single element type (e.g. alloys), since the parameter file is expected to set element interaction terms that are not captured by the information in the library file.

If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *meam* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Note

If the second filename is NULL, the element names between the two filenames can appear in any order, e.g. "Si C" or "C Si" in the example above. However, if the second filename is **not** NULL (as in the example above), it contains settings that are indexed **by numbers** for the elements that precede it. Thus you need to ensure that you list the elements between the filenames in an order consistent with how the values in the second filename are indexed. See details below on the syntax for settings in the second file.

The MEAM library file provided with LAMMPS has the name potentials/library.meam. It is the "meamf" file used by other MD codes. Aside from blank and comment lines (starting with # which can appear anywhere), it is formatted as a series of entries, each of which has 19 parameters and can span multiple lines:

elt, lat, z, ielement, atwt, alpha, b0, b1, b2, b3, alat, esub, asub, t0, t1, t2, t3, rozero, ibar

The *elt* and *lat* parameters are text strings, such as *elt* = Si or Cu and *lat* = dia or fcc. Because the library file is used by Fortran MD codes, these strings may be enclosed in single quotes, but this is not required. The other numeric

parameters match values in the formulas above. The value of the *elt* string is what is used in the pair_coeff command to identify which settings from the library file you wish to read in. There can be multiple entries in the library file with the same *elt* value; LAMMPS reads the first matching entry it finds and ignores the rest.

Other parameters in the MEAM library file correspond to single-element potential parameters:

lat = lattice structure of reference configuration
z = number of nearest neighbors in the reference structure
ielement = atomic number
atwt = atomic weight
alat = lattice constant of reference structure
esub = energy per atom (eV) in the reference structure at equilibrium
asub = "A" parameter for MEAM (see e.g. (Baskes))

The *alpha*, *b0*, *b1*, *b2*, *b3*, *t0*, *t1*, *t2*, *t3* parameters correspond to the standard MEAM parameters in the literature *(Baskes)* (the b parameters are the standard beta parameters). Note that only parameters normalized to *t0 = 1.0* are supported. The *rozero* parameter is an element-dependent density scaling that weights the reference background density (see e.g. equation 4.5 in *(Gullet)*) and is typically 1.0 for single-element systems. The *ibar* parameter selects the form of the function G(Gamma) used to compute the electron density; options are

```
 0 => G = sqrt(1+Gamma)
 1 => G = exp(Gamma/2)
 2 => not implemented
 3 => G = 2/(1+exp(-Gamma))
 4 => G = sqrt(1+Gamma)
-5 => G = +-sqrt(abs(1+Gamma))
```

If used, the MEAM parameter file contains settings that override or complement the library file settings. Examples of such parameter files are in the potentials directory with a ".meam" suffix. Their format is the same as is read by other Fortran MD codes. Aside from blank and comment lines (start with # which can appear anywhere), each line has one of the following forms. Each line can also have a trailing comment (starting with #) which is ignored.

```
keyword = value
keyword(I) = value
keyword(I,J) = value
keyword(I,J,K) = value
```

The indices I, J, K correspond to the elements selected from the MEAM library file numbered in the order of how those elements were selected starting from 1. Thus for the example given before

```
pair_coeff * * library.meam Si C sic.meam Si Si Si C
```

an index of 1 would refer to Si and an index of 2 to C.

The recognized keywords for the parameter file are as follows:

rc = cutoff radius for cutoff function; default = 4.0
delr = length of smoothing distance for cutoff function; default = 0.1
rho0(I) = relative density for element I (overwrites value read from meamf file)
Ec(I,J) = cohesive energy of reference structure for I-J mixture
delta(I,J) = heat of formation for I-J alloy; if Ec_IJ is input as zero, then LAMMPS sets Ec_IJ = (Ec_II + Ec_JJ)/2 - delta_IJ
alpha(I,J) = alpha parameter for pair potential between I and J (can be computed from bulk modulus of reference structure)
re(I,J) = equilibrium distance between I and J in the reference structure

Cmax(I,J,K) = Cmax screening parameter when I-J pair is screened
by K (I<=J); default = 2.8
Cmin(I,J,K) = Cmin screening parameter when I-J pair is screened
by K (I<=J); default = 2.0
lattce(I,J) = lattice structure of I-J reference structure:
fcc = face centered cubic
bcc = body centered cubic
hcp = hexagonal close-packed
dim = dimer
dia = diamond (interlaced fcc for alloy)
dia3= diamond structure with primary 1NN and secondary 3NN interaction
b1 = rock salt (NaCl structure)
c11 = MoSi2 structure
l12 = Cu3Au structure (lower case L, followed by 12)
b2 = CsCl structure (interpenetrating simple cubic)
ch4 = methane-like structure, only for binary system
lin = linear structure (180 degree angle)
zig = zigzag structure with a uniform angle
tri = H2O-like structure that has an angle
sc = simple cubic
nn2(I,J) = turn on second-nearest neighbor MEAM formulation for
I-J pair (see for example (Lee)).
0 = second-nearest neighbor formulation off
1 = second-nearest neighbor formulation on
default = 0
attrac(I,J) = additional cubic attraction term in Rose energy I-J pair potential
default = 0
repuls(I,J) = additional cubic repulsive term in Rose energy I-J pair potential
default = 0
zbl(I,J) = blend the MEAM I-J pair potential with the ZBL potential for small
atom separations (ZBL)
default = 1
theta(I,J) = angle between three atoms in line, zigzag, and trimer reference structures in degrees
default = 180
gsmooth_factor = factor determining the length of the G-function smoothing
region; only significant for ibar=0 or ibar=4.
99.0 = short smoothing region, sharp step
0.5 = long smoothing region, smooth step
default = 99.0
augt1 = integer flag for whether to augment t1 parameter by
3/5*t3 to account for old vs. new meam formulations;
0 = don't augment t1
1 = augment t1
default = 1
ialloy = integer flag to use alternative averaging rule for t parameters,
for comparison with the DYNAMO MEAM code
0 = standard averaging (matches ialloy=0 in DYNAMO)
1 = alternative averaging (matches ialloy=1 in DYNAMO)
2 = no averaging of t (use single-element values)
default = 0
mixture_ref_t = integer flag to use mixture average of t to compute the background
reference density for alloys, instead of the single-element values
(see description and warning elsewhere in this doc page)
0 = do not use mixture averaging for t in the reference density

1 = use mixture averaging for t in the reference density
default = 0
erose_form = integer value to select the form of the Rose energy function
(see description below).
default = 0
emb_lin_neg = integer value to select embedding function for negative densities
0 = F(rho)=0
1 = F(rho) = -asub*esub*rho (linear in rho, matches DYNAMO)
default = 0
bkgd_dyn = integer value to select background density formula
0 = rho_bkgd = rho_ref_meam(a) (as in the reference structure)
1 = rho_bkgd = rho0_meam(a)*Z_meam(a) (matches DYNAMO)
default = 0

Rc, *delr*, *re* are in distance units (Angstroms in the case of metal units). *Ec* and *delta* are in energy units (eV in the case of metal units).

Each keyword represents a quantity which is either a scalar, vector, 2d array, or 3d array and must be specified with the correct corresponding array syntax. The indices I,J,K each run from 1 to N where N is the number of MEAM elements being used.

Thus these lines

```
rho0(2) = 2.25
alpha(1,2) = 4.37
```

set *rho0* for the second element to the value 2.25 and set *alpha* for the alloy interaction between elements 1 and 2 to 4.37.

The *augt1* parameter is related to modifications in the MEAM formulation of the partial electron density function. In recent literature, an extra term is included in the expression for the third-order density in order to make the densities orthogonal (see for example *(Wang)*, equation 3d); this term is included in the MEAM implementation in LAMMPS. However, in earlier published work this term was not included when deriving parameters, including most of those provided in the library.meam file included with LAMMPS, and to account for this difference the parameter *t1* must be augmented by 3/5**t3*. If *augt1* = 1, the default, this augmentation is done automatically. When parameter values are fit using the modified density function, as in more recent literature, augt1 should be set to 0.

The *mixture_ref_t* parameter is available to match results with those of previous versions of LAMMPS (before January 2011). Newer versions of LAMMPS, by default, use the single-element values of the *t* parameters to compute the background reference density. This is the proper way to compute these parameters. Earlier versions of LAMMPS used an alloy mixture averaged value of *t* to compute the background reference density. Setting *mixture_ref_t* = 1 gives the old behavior. WARNING: using *mixture_ref_t* = 1 will give results that are demonstrably incorrect for second-neighbor MEAM, and non-standard for first-neighbor MEAM; this option is included only for matching with previous versions of LAMMPS and should be avoided if possible.

The parameters *attrac* and *repuls*, along with the integer selection parameter *erose_form*, can be used to modify the Rose energy function used to compute the pair potential. This function gives the energy of the reference state as a function of interatomic spacing. The form of this function is:

astar = alpha * (r/re - 1.d0)
if erose_form = 0: erose = -Ec*(1+astar+a3*(astar**3)/(r/re))*exp(-astar)
if erose_form = 1: erose = -Ec*(1+astar+(-attrac+repuls/r)*(astar**3))*exp(-astar)
if erose_form = 2: erose = -Ec*(1 +astar + a3*(astar**3))*exp(-astar)
a3 = repuls, astar < 0
a3 = attrac, astar >= 0

Most published MEAM parameter sets use the default values *attrac* = *repulse* = 0. Setting *repuls* = *attrac* = *delta* corresponds to the form used in several recent published MEAM parameter sets, such as *(Valone)*

Then using *meam/ms* pair style the multi-state MEAM (MS-MEAM) method is activated. This requires 6 extra parameters in the MEAM library file, resulting in 25 parameters ordered that are ordered like this:

elt, lat, z, ielement, atwt, alpha, b0, b1, b2, b3, b1m, b2m, b3m, alat, esub, asub, t0, t1, t2, t3, t1m, t2m, t3m, rozero, ibar

The 6 extra MS-MEAM parameters are *b1m, b2m, b3m, t1m, t2m, t3m*. In the LAMMPS potentials folder, compatible files have an ".msmeam" extension.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

Note

The default form of the *erose* expression in LAMMPS was corrected in March 2009. The current version is correct, but may show different behavior compared with earlier versions of LAMMPS with the attrac and/or repuls parameters are non-zero. To obtain the previous default form, use *erose_form* = 1 (this form does not seem to appear in the literature). An alternative form (see e.g. *(Lee2)*) is available using *erose_form* = 2.

4.175.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS with user-specifiable parameters as described above.

This pair style does not support the *pair_modify* *shift*, *table*, and *tail* options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.175.5 Restrictions

The *meam* and *meam/ms* pair styles are provided in the MEAM package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The maximum number of elements that can be read from the MEAM library file is determined at compile time. The default is 8. If you need support for more elements, you have to change the the constant 'MAXELT' at the beginning of the file src/MEAM/meam.h and update/recompile LAMMPS. There is no limit on the number of atoms types.

4.175.6 Related commands

pair_coeff, *pair_style eam*, *pair_style meam/spline*

4.175.7 Default

none

(Baskes) Baskes, Phys Rev B, 46, 2727-2742 (1992).

(Baskes2) Baskes, Phys Rev B, 75, 094113 (2007).

(Gullet) Gullet, Wagner, Slepoy, SANDIA Report 2003-8782 (2003). DOI:10.2172/918395 This report may be accessed on-line via this link.

(Lee) Lee, Baskes, Phys. Rev. B, 62, 8564-8567 (2000).

(Lee2) Lee, Baskes, Kim, Cho. Phys. Rev. B, 64, 184102 (2001).

(Valone) Valone, Baskes, Martin, Phys. Rev. B, 73, 214209 (2006).

(Wang) Wang, Van Hove, Ross, Baskes, J. Chem. Phys., 121, 5410 (2004).

(ZBL) J.F. Ziegler, J.P. Biersack, U. Littmark, "Stopping and Ranges of Ions in Matter", Vol 1, 1985, Pergamon Press.

4.176 pair_style meam/spline command

Accelerator Variants: *meam/spline/omp*

4.176.1 Syntax

```
pair_style meam/spline
```

4.176.2 Examples

```
pair_style meam/spline
pair_coeff * * Ti.meam.spline Ti
pair_coeff * * Ti.meam.spline Ti O
```

4.176.3 Description

The *meam/spline* style computes pairwise interactions for metals using a variant of modified embedded-atom method (MEAM) potentials *(Lenosky)*. For a single species ("old-style") MEAM, the total energy E is given by

$$E = \sum_{i<j} \phi(r_{ij}) + \sum_i U(n_i)$$
$$n_i = \sum_j \rho(r_{ij}) + \sum_{\substack{j<k,\\ j,k\neq i}} f(r_{ij}) f(r_{ik}) g[\cos(\theta_{jik})]$$

where ρ_i is the density at atom I, θ_{jik} is the angle between atoms J, I, and K centered on atom I. The five functions ϕ, U, ρ, f, and g are represented by cubic splines.

The *meam/spline* style also supports a new style multicomponent modified embedded-atom method (MEAM) potential *(Zhang)*, where the total energy E is given by

$$E = \sum_{i<j} \phi_{ij}(r_{ij}) + \sum_i U_i(n_i)$$
$$n_i = \sum_{j\neq i} \rho_j(r_{ij}) + \sum_{\substack{j<k,\\ j,k\neq i}} f_j(r_{ij}) f_k(r_{ik}) g_{jk}[\cos(\theta_{jik})]$$

where the five functions ϕ, U, ρ, f, and g depend on the chemistry of the atoms in the interaction. In particular, if there are N different chemistries, there are N different U, ρ, and f functions, while there are N(N+1)/2 different ϕ and g functions. The new style multicomponent MEAM potential files are indicated by the second line in the file starts with "meam/spline" followed by the number of elements and the name of each element.

The cutoffs and the coefficients for these spline functions are listed in a parameter file which is specified by the *pair_coeff* command. Parameter files for different elements are included in the "potentials" directory of the LAMMPS distribution and have a ".meam.spline" file suffix. All of these files are parameterized in terms of LAMMPS *metal units*.

Note that unlike for other potentials, cutoffs for spline-based MEAM potentials are not set in the pair_style or pair_coeff command; they are specified in the potential files themselves.

Unlike the EAM pair style, which retrieves the atomic mass from the potential file, the spline-based MEAM potentials do not include mass information; thus you need to use the *mass* command to specify it.

Only a single pair_coeff command is used with the *meam/spline* style which specifies a potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of spline-based MEAM elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine the Ti.meam.spline file has values for Ti (old style). In that case your LAMMPS simulation may only have one atom type which has to be mapped to the Ti element as follows:

```
pair_coeff * * Ti.meam.spline Ti
```

The first 2 arguments must be * * and there may be only one element following or NULL. Systems where there would be multiple atom types assigned to the same element are **not** supported by this pair style due to limitations in its implementation. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *meam/spline* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

An example with a two component spline (new style) is TiO.meam.spline, where the command

```
pair_coeff * * TiO.meam.spline Ti O
```

will map the first atom type to Ti and the second atom type to O. Note in this case that the species names need to match exactly with the names of the elements in the TiO.meam.spline file; otherwise an error will be raised. This behavior is different than the old style MEAM files.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.176.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

The *meam/spline* pair style does not write its information to *binary restart files*, since it is stored in an external potential parameter file. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

The *meam/spline* pair style can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.176.5 Restrictions

This pair style requires the *newton* setting to be “on” for pair interactions.

This pair style does not support mapping multiple atom types to the same element.

This pair style is only enabled if LAMMPS was built with the MANYBODY package. See the *Build package* page for more info.

4.176.6 Related commands

pair_coeff, *pair_style meam*

4.176.7 Default

none

(Lenosky) Lenosky, Sadigh, Alonso, Bulatov, de la Rubia, Kim, Voter, Kress, Modelling Simulation Materials Science Engineering, 8, 825 (2000).

(Zhang) Zhang and Trinkle, Computational Materials Science, 124, 204-210 (2016).

4.177 pair_style meam/sw/spline command

4.177.1 Syntax

```
pair_style meam/sw/spline
```

4.177.2 Examples

```
pair_style meam/sw/spline
pair_coeff * * Ti.meam.sw.spline Ti
pair_coeff * * Ti.meam.sw.spline Ti Ti Ti
```

4.177.3 Description

The *meam/sw/spline* style computes pairwise interactions for metals using a variant of modified embedded-atom method (MEAM) potentials *(Lenosky)* with an additional Stillinger-Weber (SW) term *(Stillinger)* in the energy. This form of the potential was first proposed by Nicklas, Fellinger, and Park *(Nicklas)*. We refer to it as MEAM+SW. The total energy E is given by

$$
\begin{aligned}
E &= E_{MEAM} + E_{SW} \\
E_{MEAM} &= \sum_{IJ} \phi(r_{IJ}) + \sum_{I} U(\rho_I) \\
E_{SW} &= \sum_{I} \sum_{JK} F(r_{IJ})\, F(r_{IK})\, G(\cos(\theta_{JIK})) \\
\rho_I &= \sum_{J} \rho(r_{IJ}) + \sum_{JK} f(r_{IJ})\, f(r_{IK})\, g(\cos(\theta_{JIK}))
\end{aligned}
$$

where ρ_I is the density at atom I, θ_{JIK} is the angle between atoms J, I, and K centered on atom I. The seven functions ϕ, F, G, U, ρ, f, and g are represented by cubic splines.

The cutoffs and the coefficients for these spline functions are listed in a parameter file which is specified by the *pair_coeff* command. Parameter files for different elements are included in the "potentials" directory of the LAMMPS distribution and have a ".meam.sw.spline" file suffix. All of these files are parameterized in terms of LAMMPS *metal units*.

Note that unlike for other potentials, cutoffs for spline-based MEAM+SW potentials are not set in the pair_style or pair_coeff command; they are specified in the potential files themselves.

Unlike the EAM pair style, which retrieves the atomic mass from the potential file, the spline-based MEAM+SW potentials do not include mass information; thus you need to use the *mass* command to specify it.

Only a single pair_coeff command is used with the meam/sw/spline style which specifies a potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of spline-based MEAM+SW elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine the Ti.meam.sw.spline file has values for Ti. If your LAMMPS simulation has 3 atoms types and they are all to be treated with this potential, you would use the following pair_coeff command:

```
pair_coeff * * Ti.meam.sw.spline Ti Ti Ti
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The three Ti arguments map LAMMPS atom types 1,2,3 to the Ti element in the potential file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *meam/sw/spline* potential is used as part of the hybrid pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Note

The *meam/sw/spline* style currently supports only single-element MEAM+SW potentials. It may be extended for alloy systems in the future.

Example input scripts that use this pair style are provided in the examples/PACKAGES/meam_sw_spline directory.

4.177.4 Mixing, shift, table, tail correction, restart, rRESPA info

The pair style does not support multiple element types or mixing. It has been designed for pure elements only.

This pair style does not support the *pair_modify* shift, table, and tail options.

The *meam/sw/spline* pair style does not write its information to *binary restart files*, since it is stored in an external potential parameter file. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

The *meam/sw/spline* pair style can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.177.5 Restrictions

This pair style requires the *newton* setting to be "on" for pair interactions.

This pair style is only enabled if LAMMPS was built with the MANYBODY package. See the *Build package* page for more info.

4.177.6 Related commands

pair_coeff, *pair_style meam*, *pair_style meam/spline*

4.177.7 Default

none

(Lenosky) Lenosky, Sadigh, Alonso, Bulatov, de la Rubia, Kim, Voter, Kress, Modell. Simul. Mater. Sci. Eng. 8, 825 (2000).

(Stillinger) Stillinger, Weber, Phys. Rev. B 31, 5262 (1985).

(Nicklas) The spline-based MEAM+SW format was first devised and used to develop potentials for bcc transition metals by Jeremy Nicklas, Michael Fellinger, and Hyoungki Park at The Ohio State University.

4.178 pair_style mesocnt command

4.179 pair_style mesocnt/viscous command

4.179.1 Syntax

```
pair_style style neigh_cutoff mode neigh_mode
```

- style = *mesocnt* or *mesocnt/viscous*
- neigh_cutoff = neighbor list cutoff (distance units)
- mode = *chain* or *segment* (optional)
- neigh_mode = *id* or *topology* (optional)

4.179.2 Examples

```
pair_style mesocnt 30.0
pair_coeff * * C_10_10.mesocnt 2

pair_style mesocnt/viscous 60.0 chain topology
pair_coeff * * C_10_10.mesocnt 0.001 20.0 0.2 2 4
```

4.179.3 Description

Style *mesocnt* implements a mesoscopic potential for the interaction of carbon nanotubes (CNTs), or other quasi-1D objects such as other kinds of nanotubes or nanowires. In this potential, CNTs are modelled as chains of cylindrical segments in which each infinitesimal surface element interacts with all other CNT surface elements with the Lennard-Jones (LJ) term adopted from the *airebo* style. The interaction energy is then computed by integrating over the surfaces of all interacting CNTs.

In LAMMPS, cylindrical segments are represented by bonds. Each segment is defined by its two end points ("nodes") which correspond to atoms in LAMMPS. For the exact functional form of the potential and implementation details, the reader is referred to the original papers *(Volkov1)* and *(Volkov2)*.

Changed in version 15Sep2022.

The potential supports two modes, *segment* and *chain*. By default, *chain* mode is enabled. In *segment* mode, interactions are pair-wise between all neighboring segments based on a segment-segment approach (keyword *segment* in pair_style command). In *chain* mode, interactions are calculated between each segment and infinitely or semi-infinitely long CNTs as described in *(Volkov1)*. Chains of segments are converted to these (semi-)infinite CNTs bases on an approximate chain approach outlined in *(Volkov2)*. Hence, interactions are calculated on a segment-chain basis (keyword *chain* in the pair_style command). Using *chain* mode allows to simplify the computation of the interactions significantly and reduces the computational times to the same order of magnitude as for regular bead spring models where beads interact with the standard *pair_lj/cut* potential. However, this method is only valid when the curvature of the CNTs in the system is small. When CNTs are buckled (see *angle_mesocnt*), local curvature can be very high and the pair_style automatically switches to *segment* mode for interactions involving buckled CNTs.

The potential further implements two different neighbor list construction modes. Mode *id* uses atom and mol IDs to construct neighbor lists while *topology* modes uses only the bond topology of the system. While *id* mode requires bonded atoms to have consecutive LAMMPS atom IDs and atoms in different CNTs to have different LAMMPS molecule IDs, *topology* mode has no such requirement. Using *id* mode is faster and is enabled by default.

Note

Neighbor *id* mode requires all CNTs in the system to have distinct LAMMPS molecule IDs and bonded atoms to have consecutive LAMMPS atom IDs. If this is not possible (e.g. in simulations of CNT rings), *topology* mode needs to be enabled in the pair_style command.

Added in version 15Sep2022.

In addition to the LJ interactions described above, style *mesocnt/viscous* explicitly models friction between neighboring segments. Friction forces are a function of the relative velocity between a segment and its neighboring approximate chain (even in *segment* mode) and only act along the axes of the interacting segment and chain. In this potential, friction forces acting per unit length of a nanotube segment are modelled as a shifted logistic function:

$$F^{\mathrm{FRICTION}}(v)/L = \frac{F^{\max}}{1+\exp(-k(v-v_0))} - \frac{F^{\max}}{1+\exp(kv_0)}$$

In the pair_style command, the modes described above can be toggled using the *segment* or *chain* keywords. The neighbor list cutoff defines the cutoff within which atoms are included in the neighbor list for constructing neighboring CNT chains. This is different from the potential cutoff, which is directly calculated from parameters specified in the potential file. We recommend using a neighbor list cutoff of at least 3 times the maximum segment length used in the simulation to ensure proper neighbor chain construction.

Note

CNT ends are treated differently by all *mesocnt* styles. Atoms on CNT ends need to be assigned different LAMMPS atom types than atoms not on CNT ends.

Style *mesocnt* requires tabulated data provided in a single ASCII text file, as well as a list of integers corresponding to all LAMMPS atom types representing CNT ends:

- filename

- N CNT end atom types

For example, if your LAMMPS simulation of (10, 10) nanotubes has 4 atom types where atom types 1 and 3 are assigned to 'inner' nodes and atom types 2 and 4 are assigned to CNT end nodes, the pair_coeff command would be:

```
pair_coeff * * C_10_10.mesocnt 2 4
```

Likewise, style *mesocnt/viscous* also requires the same information as style *mesocnt*, with the addition of 3 parameters for the viscous friction forces as listed above:

- filename
- $F^{\max}$
- k
- v_0
- N CNT end atom types

Using the same example system as with style *mesocnt* with the addition of friction, the pair_coeff command is:

```
pair_coeff * * C_10_10.mesocnt 0.03 20.0 0.20 2 4
```

Potential files for CNTs can be readily generated using the freely available code provided on

```
https://github.com/phankl/cntpot
```

Using the same approach, it should also be possible to generate potential files for other 1D systems mentioned above.

Note

Because of their size, *mesocnt* style potential files are not bundled with LAMMPS. When compiling LAMMPS from source code, the file C_10_10.mesocnt should be downloaded separately from https://download.lammps.org/potentials/C_10_10.mesocnt

The first line of the potential file provides a time stamp and general information. The second line lists four integers giving the number of data points provided in the subsequent four data tables. The third line lists four floating point numbers: the CNT radius R, the LJ parameter sigma and two numerical parameters delta1 and delta2. These four parameters are given in Angstroms. This is followed by four data tables each separated by a single empty line. The first two tables have two columns and list the parameters uInfParallel and Gamma respectively. The last two tables have three columns giving data on a quadratic array and list the parameters Phi and uSemiParallel respectively. uInfParallel and uSemiParallel are given in eV/Angstrom, Phi is given in eV and Gamma is unitless.

If a simulation produces many warnings about segment-chain interactions falling outside the interpolation range, we recommend generating a potential file with lower values of delta1 and delta2.

4.179.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles does not support mixing.

These pair styles does not support the *pair_modify* shift, table, and tail options.

These pair styles do not write their information to *binary restart files*, since it is stored in tabulated potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.179.5 Restrictions

These styles are part of the MESONT package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles require the *newton* setting to be "on" for pair interactions.

These pair styles require all 3 *special_bonds lj* settings to be non-zero for proper neighbor list construction.

Pair style *mesocnt/viscous* requires you to use the *comm_modify vel yes* command so that velocities are stored by ghost atoms.

4.179.6 Related commands

pair_coeff, *bond_style mesocnt*, *angle_style mesocnt*

4.179.7 Default

mode = chain, neigh_mode = id

(Volkov1) Volkov and Zhigilei, J Phys Chem C, 114, 5513 (2010).

(Volkov2) Volkov, Simov and Zhigilei, APS Meeting Abstracts, Q31.013 (2008).

4.180 pair_style edpd command

Accelerator Variants: *edpd/gpu*

4.181 pair_style mdpd command

Accelerator Variants: *mdpd/gpu*

4.182 pair_style mdpd/rhosum command

4.183 pair_style tdpd command

4.183.1 Syntax

```
pair_style style args
```

- style = *edpd* or *mdpd* or *mdpd/rhosum* or *tdpd*
- args = list of arguments for a particular style

edpd args = cutoff seed
  cutoff = global cutoff for eDPD interactions (distance units)
  seed = random # seed (integer) (if <= 0, eDPD will use current time as the seed)
mdpd args = T cutoff seed
  T = temperature (temperature units)
  cutoff = global cutoff for mDPD interactions (distance units)
  seed = random # seed (integer) (if <= 0, mDPD will use current time as the seed)
mdpd/rhosum args =
tdpd args = T cutoff seed
  T = temperature (temperature units)
  cutoff = global cutoff for tDPD interactions (distance units)
  seed = random # seed (integer) (if <= 0, tDPD will use current time as the seed)

4.183.2 Examples

```
pair_style edpd 1.58 9872598
pair_coeff * * 18.75 4.5 0.41 1.58 1.42E-5 2.0 1.58
pair_coeff 1 1 18.75 4.5 0.41 1.58 1.42E-5 2.0 1.58 power 10.54 -3.66 3.44 -4.10
pair_coeff 1 1 18.75 4.5 0.41 1.58 1.42E-5 2.0 1.58 power 10.54 -3.66 3.44 -4.10 kappa -0.44 -3.21 5.04 0.00

pair_style hybrid/overlay mdpd/rhosum mdpd 1.0 1.0 65689
pair_coeff 1 1 mdpd/rhosum  0.75
pair_coeff 1 1 mdpd -40.0 25.0 18.0 1.0 0.75

pair_style tdpd 1.0 1.58 935662
pair_coeff * * 18.75 4.5 0.41 1.58 1.58 1.0 1.0E-5 2.0
pair_coeff 1 1 18.75 4.5 0.41 1.58 1.58 1.0 1.0E-5 2.0 3.0 1.0E-5 2.0
```

4.183.3 Description

The *edpd* style computes the pairwise interactions and heat fluxes for eDPD particles following the formulations in *(Li2014_JCP)* and *Li2015_CC*. The time evolution of an eDPD particle is governed by the conservation of momentum and energy given by

$$\frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} = \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{F}_i = \sum_{i \neq j} (\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R)$$

$$C_v \frac{\mathrm{d}T_i}{\mathrm{d}t} = q_i = \sum_{i \neq j} (q_{ij}^C + q_{ij}^V + q_{ij}^R),$$

where the three components of F_i including the conservative force F_{ij}^C, dissipative force F_{ij}^D and random force F_{ij}^R are expressed as

$$\begin{aligned}
\mathbf{F}_{ij}^C &= \alpha_{ij}\omega_C(r_{ij})\mathbf{e}_{ij} \\
\mathbf{F}_{ij}^D &= -\gamma\omega_D(r_{ij})(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\mathbf{e}_{ij} \\
\mathbf{F}_{ij}^R &= \sigma\omega_R(r_{ij})\xi_{ij}\Delta t^{-1/2}\mathbf{e}_{ij} \\
\omega_C(r) &= 1 - r/r_c \\
\alpha_{ij} &= A\cdot k_B(T_i+T_j)/2 \\
\omega_D(r) &= \omega_R^2(r) = (1-r/r_c)^s \\
\sigma_{ij}^2 &= 4\gamma k_B T_i T_j/(T_i+T_j)
\end{aligned}$$

in which the exponent of the weighting function s can be defined as a temperature-dependent variable. The heat flux between particles accounting for the collisional heat flux q^C, viscous heat flux q^V, and random heat flux q^R are given by

$$\begin{aligned}
q_i^C &= \sum_{j\neq i} k_{ij}\omega_{CT}(r_{ij})\left(\frac{1}{T_i}-\frac{1}{T_j}\right) \\
q_i^V &= \frac{1}{2C_v}\sum_{j\neq i}\left\{\omega_D(r_{ij})\left[\gamma_{ij}\left(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij}\right)^2 - \frac{\left(\sigma_{ij}\right)^2}{m}\right] - \sigma_{ij}\omega_R(r_{ij})\left(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij}\right)\xi_{ij}\right\} \\
q_i^R &= \sum_{j\neq i}\beta_{ij}\omega_{RT}(r_{ij})dt^{-1/2}\xi_{ij}^e \\
\omega_{CT}(r) &= \omega_{RT}^2(r) = \left(1-r/r_{ct}\right)^{s_T} \\
k_{ij} &= C_v^2\kappa(T_i+T_j)^2/4k_B \\
\beta_{ij}^2 &= 2k_B k_{ij}
\end{aligned}$$

where the mesoscopic heat friction κ is given by

$$\kappa = \frac{315k_B\upsilon}{2\pi\rho C_v r_{ct}^5}\frac{1}{Pr},$$

with υ being the kinematic viscosity. For more details, see Eq.(15) in *(Li2014_JCP)*.

The following coefficients must be defined in eDPD system for each pair of atom types via the *pair_coeff* command as in the examples above.

- A (force units)
- γ (force/velocity units)
- power_f (positive real)
- cutoff (distance units)
- kappa (thermal conductivity units)
- power_T (positive real)
- cutoff_T (distance units)
- optional keyword = power or kappa

The keyword *power* or *kappa* is optional. Both "power" and "kappa" require 4 parameters c_1, c_2, c_3, c_4 showing the temperature dependence of the exponent $s(T) = \text{power}_f(1+c_1(T-1)+c_2(T-1)^2+c_3(T-1)^3+c_4(T-1)^4)$ and of the mesoscopic heat friction $s_T(T) = \kappa(1+c_1(T-1)+c_2(T-1)^2+c_3(T-1)^3+c_4(T-1)^4)$. If the keyword *power* or *kappa* is not specified, the eDPD system will use constant power_f and κ, which is independent to temperature changes.

The *mdpd/rhosum* style computes the local particle mass density ρ for mDPD particles by kernel function interpolation.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above.

- cutoff (distance units)

The *mdpd* style computes the many-body interactions between mDPD particles following the formulations in *(Li2013_POF)*. The dissipative and random forces are in the form same as the classical DPD, but the conservative force is local density dependent, which are given by

$$\begin{aligned}\mathbf{F}_{ij}^{C} &= Aw_c(r_{ij})\mathbf{e}_{ij} + B(\rho_i + \rho_j)w_d(r_{ij})\mathbf{e}_{ij}\\ \mathbf{F}_{ij}^{D} &= -\gamma\omega_D(r_{ij})(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\mathbf{e}_{ij}\\ \mathbf{F}_{ij}^{R} &= \sigma\omega_R(r_{ij})\xi_{ij}\Delta t^{-1/2}\mathbf{e}_{ij}\end{aligned}$$

where the first term in F_C with a negative coefficient $A < 0$ stands for an attractive force within an interaction range r_c, and the second term with $B > 0$ is the density-dependent repulsive force within an interaction range r_d.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above.

- A (force units)
- B (force units)
- γ (force/velocity units)
- cutoff_c (distance units)
- cutoff_d (distance units)

The *tdpd* style computes the pairwise interactions and chemical concentration fluxes for tDPD particles following the formulations in *(Li2015_JCP)*. The time evolution of a tDPD particle is governed by the conservation of momentum and concentration given by

$$\begin{aligned}\frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} = \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{F}_i = \sum_{i\neq j}(\mathbf{F}_{ij}^{C} + \mathbf{F}_{ij}^{D} + \mathbf{F}_{ij}^{R})\\ \frac{\mathrm{d}C_i}{\mathrm{d}t} = Q_i = \sum_{i\neq j}(Q_{ij}^{D} + Q_{ij}^{R}) + Q_i^{S}\end{aligned}$$

where the three components of F_i including the conservative force F_{ij}^C, dissipative force F_{ij}^C and random force F_{ij}^C are expressed as

$$\begin{aligned}\mathbf{F}_{ij}^{C} &= A\omega_C(r_{ij})\mathbf{e}_{ij}\\ \mathbf{F}_{ij}^{D} &= -\gamma\omega_D(r_{ij})(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\mathbf{e}_{ij}\\ \mathbf{F}_{ij}^{R} &= \sigma\omega_R(r_{ij})\xi_{ij}\Delta t^{-1/2}\mathbf{e}_{ij}\\ \omega_C(r) &= 1 - r/r_c\\ \omega_D(r) &= \omega_R^2(r) = (1 - r/r_c)^{\mathrm{power_f}}\\ \sigma^2 &= 2\gamma k_B T\end{aligned}$$

The concentration flux between two tDPD particles includes the Fickian flux Q_{ij}^D and random flux Q_{ij}^R, which are given by

$$
\begin{aligned}
Q_{ij}^D &= -\kappa_{ij} w_{DC}(r_{ij}) \left(C_i - C_j\right) \\
Q_{ij}^R &= \varepsilon_{ij} \left(C_i + C_j\right) w_{RC}(r_{ij}) \xi_{ij} \\
w_{DC}(r_{ij}) &= w_{RC}^2(r_{ij}) = (1 - r/r_{cc})^{\text{power}_{cc}} \\
\varepsilon_{ij}^2 &= m_s^2 \kappa_{ij} \rho
\end{aligned}
$$

where the parameters kappa and epsilon determine the strength of the Fickian and random fluxes. m_s is the mass of a single solute molecule. In general, m_s is much smaller than the mass of a tDPD particle m. For more details, see *(Li2015_JCP)*.

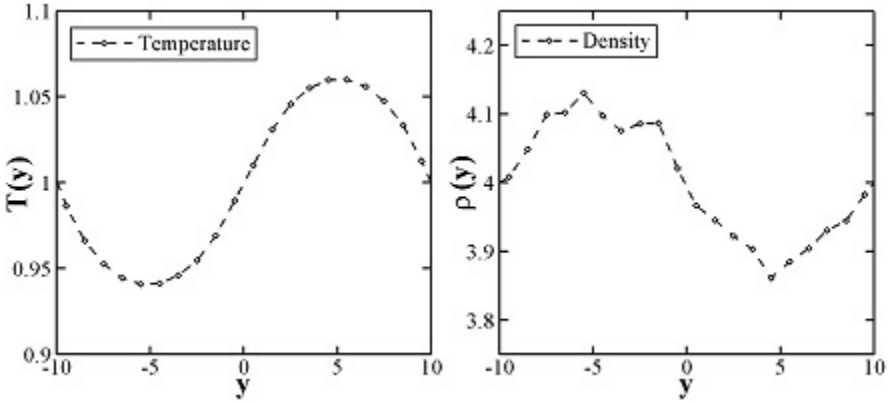
The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above.

- A (force units)
- γ (force/velocity units)
- power_f (positive real)
- cutoff (distance units)
- cutoff_CC (distance units)
- κ_i (diffusivity units)
- ε_i (diffusivity units)
- power_cc_i (positive real)

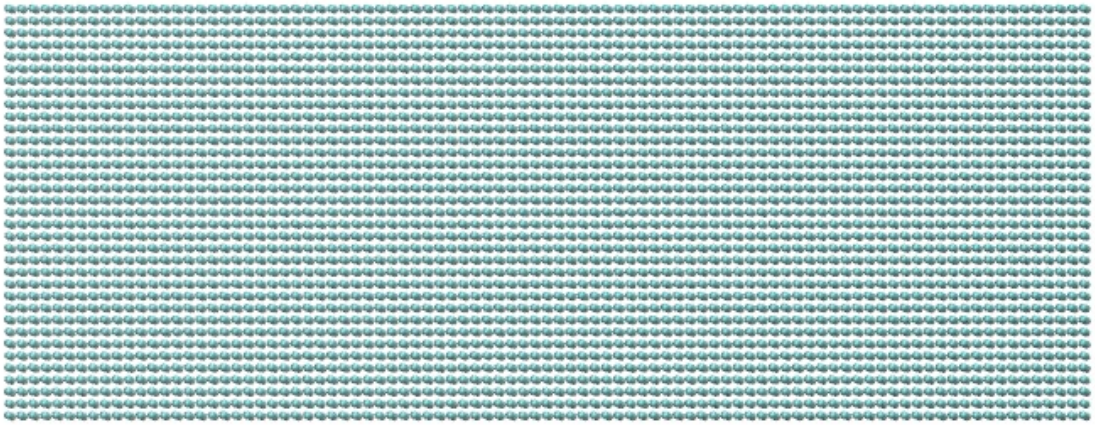
The last 3 values must be repeated Nspecies times, so that values for each of the Nspecies chemical species are specified, as indicated by the "I" suffix. In the first pair_coeff example above for pair_style tdpd, Nspecies = 1. In the second example, Nspecies = 2, so 3 additional coeffs are specified (for species 2).

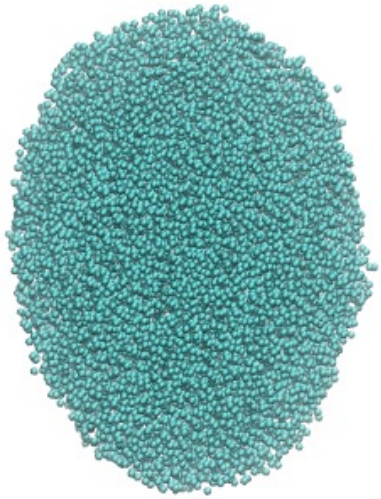
4.183.4 Example scripts

There are example scripts for using all these pair styles in examples/PACKAGES/mesodpd. The example for an eDPD simulation models heat conduction with source terms analog of periodic Poiseuille flow problem. The setup follows Fig.12 in *(Li2014_JCP)*. The output of the short eDPD simulation (about 2 minutes on a single core) gives a temperature and density profiles as



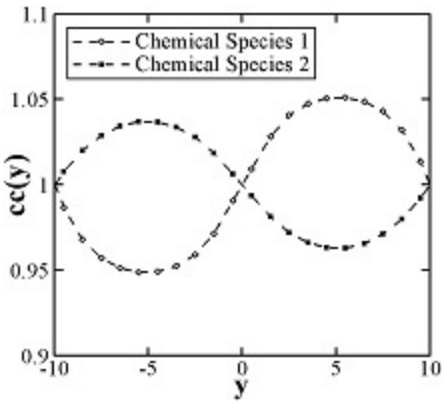
The example for a mDPD simulation models the oscillations of a liquid droplet started from a liquid film. The mDPD parameters are adopted from *(Li2013_POF)*. The short mDPD run (about 2 minutes on a single core) generates a particle trajectory which can be visualized as follows.





The first image is the initial state of the simulation. If you click it a GIF movie should play in your browser. The second image is the final state of the simulation.

The example for a tDPD simulation computes the effective diffusion coefficient of a tDPD system using a method analogous to the periodic Poiseuille flow. The tDPD system is specified with two chemical species, and the setup follows Fig.1 in *(Li2015_JCP)*. The output of the short tDPD simulation (about one and a half minutes on a single core) gives the concentration profiles of the two chemical species as



4.183.5 Mixing, shift, table, tail correction, restart, rRESPA info

The styles *edpd*, *mdpd*, *mdpd/rhosum* and *tdpd* do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The styles *edpd*, *mdpd*, *mdpd/rhosum* and *tdpd* do not support the *pair_modify* shift, table, and tail options.

The styles *edpd*, *mdpd*, *mdpd/rhosum* and *tdpd* do not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.183.6 Restrictions

The pair styles *edpd*, *mdpd*, *mdpd/rhosum* and *tdpd* are part of the DPD-MESO package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.183.7 Related commands

pair_coeff, *fix mvv/dpd*, *fix mvv/edpd*, *fix mvv/tdpd*, *fix edpd/source*, *fix tdpd/source*, *compute edpd/temp/atom*, *compute tdpd/cc/atom*

4.183.8 Default

none

(Li2014_JCP) Li, Tang, Lei, Caswell, Karniadakis, J Comput Phys, 265: 113-127 (2014). DOI: 10.1016/j.jcp.2014.02.003.

(Li2015_CC) Li, Tang, Li, Karniadakis, Chem Commun, 51: 11038-11040 (2015). DOI: 10.1039/C5CC01684C.

(Li2013_POF) Li, Hu, Wang, Ma, Zhou, Phys Fluids, 25: 072103 (2013). DOI: 10.1063/1.4812366.

(Li2015_JCP) Li, Yazdani, Tartakovsky, Karniadakis, J Chem Phys, 143: 014101 (2015). DOI: 10.1063/1.4923254.

4.184 pair_style mgpt command

4.184.1 Syntax

```
pair_style mgpt
```

4.184.2 Examples

```
pair_style mgpt
pair_coeff * * Ta6.8x.mgpt.parmin Ta6.8x.mgpt.potin Omega
cp ~/lammps/potentials/Ta6.8x.mgpt.parmin parmin
cp ~/lammps/potentials/Ta6.8x.mgpt.potin potin
pair_coeff * * parmin potin Omega volpress yes nbody 1234 precision double
pair_coeff * * parmin potin Omega volpress yes nbody 12
```

4.184.3 Description

Within DFT quantum mechanics, generalized pseudopotential theory (GPT) (*Moriarty1*) provides a first-principles approach to multi-ion interatomic potentials in d-band transition metals, with a volume-dependent, real-space total-energy functional for the N-ion elemental bulk material in the form

$$E_{\rm tot}(\mathbf{R}_1 \ldots \mathbf{R}_N) = NE_{\rm vol}(\Omega) + \frac{1}{2}\sum_{i,j}{}' \, v_2(ij;\Omega) + \frac{1}{6}\sum_{i,j,k}{}' \, v_3(ijk;\Omega) + \frac{1}{24}\sum_{i,j,k,l}{}' \, v_4(ijkl;\Omega)$$

where the prime on each summation sign indicates the exclusion of all self-interaction terms from the summation. The leading volume term E_vol as well as the two-ion central-force pair potential v_2 and the three- and four-ion angular-force potentials, v_3 and v_4, depend explicitly on the atomic volume Omega, but are structure independent and transferable to all bulk ion configurations, either ordered or disordered, and with of without the presence of point and line defects. The simplified model GPT or MGPT (*Moriarty2*, *Moriarty3*), which retains the form of E_tot and permits more efficient large-scale atomistic simulations, derives from the GPT through a series of systematic approximations applied to E_vol and the potentials v_n that are valid for mid-period transition metals with nearly half-filled d bands.

Both analytic (*Moriarty2*) and matrix (*Moriarty3*) representations of MGPT have been developed. In the more general matrix representation, which can also be applied to f-band actinide metals and permits both canonical and non-canonical d/f bands, the multi-ion potentials are evaluated on the fly during a simulation through d- or f-state matrix multiplication, and the forces that move the ions are determined analytically. Fast matrix-MGPT algorithms have been developed independently by Glosli (*Glosli*, *Moriarty3*) and by Oppelstrup (*Oppelstrup*)

The *mgpt* pair style calculates forces, energies, and the total energy per atom, E_tot/N, using the Oppelstrup matrix-MGPT algorithm. Input potential and control data are entered through the *pair_coeff* command. Each material treated requires input parmin and potin potential files, as shown in the above examples, as well as specification by the user of the initial atomic volume Omega through pair_coeff. At the beginning of a time step in any simulation, the total volume of the simulation cell V should always be equal to Omega*N, where N is the number of metal ions present, taking into account the presence of any vacancies and/or interstitials in the case of a solid. In a constant-volume simulation, which is the normal mode of operation for the *mgpt* pair style, Omega, V and N all remain constant throughout the simulation and thus are equal to their initial values. In a constant-stress simulation, the cell volume V will change (slowly) as the simulation proceeds. After each time step, the atomic volume should be updated by the code as Omega = V/N. In addition, the volume term E_vol and the potentials v_2, v_3 and v_4 have to be removed at the end of the time step, and then respecified at the new value of Omega. In all simulations, Omega must remain within the defined volume range for E_vol and the potentials for the given material.

The default option volpress yes in the *pair_coeff* command includes all volume derivatives of E_tot required to calculate the stress tensor and pressure correctly. The option volpress no disregards the pressure contribution resulting from the volume term E_vol, and can be used for testing and analysis purposes. The additional optional variable nbody controls the specific terms in E_tot that are calculated. The default option and the normal option for mid-period transition and actinide metals is nbody 1234 for which all four terms in E_tot are retained. The option nbody 12, for example, retains only the volume term and the two-ion pair potential term and can be used for GPT series-end transition metals that can be well described without v_3 and v_4. The nbody option can also be used to test or analyze the contribution of any of the four terms in E_tot to a given calculated property.

The *mgpt* pair style makes extensive use of matrix algebra and includes optimized kernels for the BlueGene/Q architecture and the Intel/AMD (x86) architectures. When compiled with the appropriate compiler and compiler switches (-msse3 on x86, and using the IBM XL compiler on BG/Q), these optimized routines are used automatically. For BG/Q machines, building with the default Makefile for that architecture (e.g., "make bgq") should enable the optimized algebra routines. For x-86 machines, there is a provided Makefile.mgptfast which enables the fast algebra routines, i.e. build LAMMPS with "make mgptfast". The user will be informed in the output files of the matrix kernels in use. To further improve speed, on x86 the option precision single can be added to the *pair_coeff* command line, which improves speed (up to a factor of two) at the cost of doing matrix calculations with 7 digit precision instead of the default 16. For consistency the default option can be specified explicitly by the option precision double.

All remaining potential and control data are contained with the parmin and potin files, including cutoffs, atomic mass, and other basic MGPT variables. Specific MGPT potential data for the transition metals tantalum (Ta4 and Ta6.8x potentials), molybdenum (Mo5.2 potentials), and vanadium (V6.1 potentials) are contained in the LAMMPS potentials directory. The stored files are, respectively, Ta4.mgpt.parmin, Ta4.mgpt.potin, Ta6.8x.mgpt.parmin, Ta6.8x.mgpt.potin, Mo5.2.mgpt.parmin, Mo5.2.mgpt.potin, V6.1.mgpt.parmin, and V6.1.mgpt.potin . Useful corresponding informational "README" files on the Ta4, Ta6.8x, Mo5.2 and V6.1 potentials are also included in the potentials directory. These latter files indicate the volume mesh and range for each potential and give appropriate references for the potentials. It is expected that MGPT potentials for additional materials will be added over time.

Useful example MGPT scripts are given in the examples/PACKAGES/mgpt directory. These scripts show the necessary steps to perform constant-volume calculations and simulations. It is strongly recommended that the user work through and understand these examples before proceeding to more complex simulations.

Note

For good performance, LAMMPS should be built with the compiler flags "-O3 -msse3 -funroll-loops" when including this pair style. The src/MAKE/OPTIONS/Makefile.mgptfast is an example machine Makefile with these options included as part of a standard MPI build. Note that it as provided, it will build with whatever low-level compiler (g++, icc, etc) is the default for your MPI installation.

4.184.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you needs to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.184.5 Restrictions

This pair style is part of the MGPT package and is only enabled if LAMMPS is built with that package. See the *Build package* page for more info.

The MGPT potentials require the *newton* setting to be "on" for pair style interactions.

The stored parmin and potin potential files provided with LAMMPS in the "potentials" directory are written in Rydberg atomic units, with energies in Rydbergs and distances in Bohr radii. The *mgpt* pair style converts Rydbergs to Hartrees to make the potential files compatible with LAMMPS electron *units*.

The form of E_tot used in the *mgpt* pair style is only appropriate for elemental bulk solids and liquids. This includes solids with point and extended defects such as vacancies, interstitials, grain boundaries and dislocations. Alloys and free surfaces, however, require significant modifications, which are not included in the *mgpt* pair style. Likewise, the *hybrid* pair style is not allowed, where MGPT would be used for some atoms but not for others.

Electron-thermal effects are not included in the standard MGPT potentials provided in the "potentials" directory, where the potentials have been constructed at zero electron temperature. Physically, electron-thermal effects may be important in 3d (e.g., V) and 4d (e.g., Mo) transition metals at high temperatures near melt and above. It is expected that temperature-dependent MGPT potentials for such cases will be added over time.

4.184.6 Related commands

pair_coeff

4.184.7 Default

The options defaults for the *pair_coeff* command are volpress yes, nbody 1234, and precision double.

(Moriarty1) Moriarty, Physical Review B, 38, 3199 (1988).

(Moriarty2) Moriarty, Physical Review B, 42, 1609 (1990). Moriarty, Physical Review B 49, 12431 (1994).

(Moriarty3) Moriarty, Benedict, Glosli, Hood, Orlikowski, Patel, Soderlind, Streitz, Tang, and Yang, Journal of Materials Research, 21, 563 (2006).

(Glosli) Glosli, unpublished, 2005. Streitz, Glosli, Patel, Chan, Yates, de Supinski, Sexton and Gunnels, Journal of Physics: Conference Series, 46, 254 (2006).

(Oppelstrup) Oppelstrup, unpublished, 2015. Oppelstrup and Moriarty, to be published.

4.185 pair_style mie/cut command

Accelerator Variants: *mie/cut/gpu*

4.185.1 Syntax

```
pair_style mie/cut cutoff
```

- cutoff = global cutoff for mie/cut interactions (distance units)

4.185.2 Examples

```
pair_style mie/cut 10.0
pair_coeff 1 1 0.72 3.40 23.00 6.66
pair_coeff 2 2 0.30 3.55 12.65 6.00
pair_coeff 1 2 0.46 3.32 16.90 6.31
```

4.185.3 Description

The *mie/cut* style computes the Mie potential, given by

$$E = C\varepsilon \left[\left(\frac{\sigma}{r}\right)^{\gamma_{rep}} - \left(\frac{\sigma}{r}\right)^{\gamma_{att}} \right] \qquad r < r_c$$

r_c is the cutoff and C is a function that depends on the repulsive and attractive exponents, given by:

$$C = \left(\frac{\gamma_{rep}}{\gamma_{rep} - \gamma_{att}}\right) \left(\frac{\gamma_{rep}}{\gamma_{att}}\right)^{\left(\frac{\gamma_{att}}{\gamma_{rep} - \gamma_{att}}\right)}$$

Note that for 12/6 exponents, C is equal to 4 and the formula is the same as the standard Lennard-Jones potential.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- epsilon (energy units)
- sigma (distance units)
- gammaR
- gammaA
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used.

4.185.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the mie/cut pair styles can be mixed. If not explicitly defined, both the repulsive and attractive gamma exponents for different atoms will be calculated following the same mixing rule defined for distances. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

This pair style supports the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure of the pair interaction.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style supports the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command, meaning the pairwise forces can be partitioned by distance at different levels of the rRESPA hierarchy. See the *run_style* command for details.

4.185.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.185.6 Related commands

pair_coeff

4.185.7 Default

none

(Mie) G. Mie, Ann Phys, 316, 657 (1903).

(Avendano) C. Avendano, T. Lafitte, A. Galindo, C. S. Adjiman, G. Jackson, E. Muller, J Phys Chem B, 115, 11154 (2011).

4.186 pair_style mliap command

Accelerator Variants: *mliap/kk*

4.186.1 Syntax

```
pair_style mliap ... keyword values ...
```

- one or two keyword/value pairs must be appended
- keyword = *model* or *descriptor* or *unified*

```
model values = style filename
  style = linear or quadratic or nn or mliappy
  filename = name of file containing model definitions
descriptor values = style filename
  style = sna or so3 or ace
  filename = name of file containing descriptor definitions
unified values = filename ghostneigh_flag
  filename = name of file containing serialized unified Python object
  ghostneigh_flag = 0/1 to turn off/on inclusion of ghost neighbors in neighbors list
```

4.186.2 Examples

```
pair_style mliap model linear InP.mliap.model descriptor sna InP.mliap.descriptor
pair_style mliap model quadratic W.mliap.model descriptor sna W.mliap.descriptor
pair_style mliap model nn Si.nn.mliap.model descriptor so3 Si.nn.mliap.descriptor
pair_style mliap model mliappy ACE_NN_Pytorch.pt descriptor ace ccs_single_element.yace
pair_style mliap unified mliap_unified_lj_Ar.pkl 0
pair_coeff * * In P
```

4.186.3 Description

Pair style *mliap* provides a general interface to families of machine-learning interatomic potentials. It allows separate definitions of the interatomic potential functional form (*model*) and the geometric quantities that characterize the atomic positions (*descriptor*).

By defining *model* and *descriptor* separately, it is possible to use many different models with a given descriptor, or many different descriptors with a given model. The pair style currently supports *sna*, *so3* and *ace* descriptor styles, but it is straightforward to add new descriptor styles. By using the *unified* keyword, it is possible to define a Python model that combines functionalities of both *model* and *descriptor*.

The SNAP descriptor style *sna* is the same as that used by *pair_style snap*, including the linear, quadratic, and chem variants. The available models are *linear*, *quadratic*, *nn*, and *mliappy*. The *mliappy* style can be used to couple python models, e.g. PyTorch neural network energy models, and requires building LAMMPS with the PYTHON package (see below). In order to train a model, it is useful to know the gradient or derivative of energy, force, and stress w.r.t. model parameters. This information can be accessed using the related *compute mliap* command.

Added in version 2Jun2022.

The descriptor style *so3* is a descriptor that is derived from the the smooth SO(3) power spectrum with the explicit inclusion of a radial basis *(Bartok)* and *(Zagaceta)*. The available models are *linear* and *nn*.

Added in version 17Apr2024.

The descriptor style *ace* is a class of highly general atomic descriptors, atomic cluster expansion descriptors (ACE) from *(Drautz)*, that include a radial basis, an angular basis, and bases for other variables (such as chemical species) if relevant. In descriptor style *ace*, the *ace* descriptors may be defined up to an arbitrary body order. This descriptor style is the same as that used in *pair_style pace* and *compute pace*. The available models with *ace* in ML-IAP are *linear* and *mliappy*. The *ace* descriptors and models require building LAMMPS with the ML-PACE package (see below). The *mliappy* model style may be used with *ace* descriptors, but it requires that LAMMPS is also built with the PYTHON package. As with other model styles, the *mliappy* model style can be used to couple arbitrary python models that use the *ace* descriptors such as Pytorch NNs. Note that *ALL* mliap model styles with *ace* descriptors require that descriptors and hyperparameters are supplied in a *.yace* or *.ace* file, similar to *compute pace*.

The pair_style *mliap* command must be followed by two keywords *model* and *descriptor* in either order, or the one keyword *unified*. A single *pair_coeff* command is also required. The first 2 arguments must be * * so as to span all LAMMPS atom types. This is followed by a list of N arguments that specify the mapping of MLIAP element names to LAMMPS atom types, where N is the number of LAMMPS atom types.

The *model* keyword is followed by the model style. This is followed by a single argument specifying the model filename containing the parameters for a set of elements. The model filename usually ends in the *.mliap.model* extension. It may contain parameters for many elements. The only requirement is that it contain at least those element names appearing in the *pair_coeff* command.

The top of the model file can contain any number of blank and comment lines (start with #), but follows a strict format after that. The first non-blank non-comment line must contain two integers:

- nelems = Number of elements

- nparams = Number of parameters

When the *model* keyword is *linear* or *quadratic*, this is followed by one block for each of the *nelem* elements. Each block consists of *nparams* parameters, one per line. Note that this format is similar, but not identical to that used for the *pair_style snap* coefficient file. Specifically, the line containing the element weight and radius is omitted, since these are handled by the *descriptor*.

When the *model* keyword is *nn* (neural networks), the model file can contain blank and comment lines (start with #) anywhere. The second non-blank non-comment line must contain the string NET, followed by two integers:

- ndescriptors = Number of descriptors
- nlayers = Number of layers (including the hidden layers and the output layer)

and followed by a sequence of a string and an integer for each layer:

- Activation function (linear, sigmoid, tanh or relu)
- nnodes = Number of nodes

This is followed by one block for each of the *nelem* elements. Each block consists of *scale0* minimum value, *scale1* (maximum - minimum) value, in order to normalize the descriptors, followed by *nparams* parameters, including *bias* and *weights* of the model, starting with the first node of the first layer and so on, with a maximum of 30 values per line.

The detail of *nn* module implementation can be found at *(Yanxon)*.

Notes on mliappy models

When the *model* keyword is *mliappy*, if the filename ends in '.pt', or '.pth', it will be loaded using pytorch; otherwise, it will be loaded as a pickle file. To load a model from memory (i.e. an existing python object), specify the filename as "LATER", and then call *lammps.mliap.load_model(model)* from python before using the pair style. When using LAMMPS via the library mode, you will need to call *lammps.mliappy.activate_mliappy(lmp)* on the active LAMMPS object before the pair style is defined. This call locates and loads the mliap-specific python module that is built into LAMMPS.

The *descriptor* keyword is followed by a descriptor style, and additional arguments. Currently three descriptor styles are available: *sna*, *so3*, and *ace*.

- *sna* indicates the bispectrum component descriptors used by the Spectral Neighbor Analysis Potential (SNAP) potentials of *pair_style snap*. A single additional argument specifies the descriptor filename containing the parameters and setting used by the SNAP descriptor. The descriptor filename usually ends in the *.mliap.descriptor* extension.
- *so3* indicated the power spectrum component descriptors. A single additional argument specifies the descriptor filename containing the parameters and setting.
- *ace* indicates the atomic cluster expansion (ACE) descriptors. A single additional argument specifies the filename containing parameters, settings, and definitions of the ace descriptors (through tabulated basis function indices and corresponding generalized Clebsch-Gordan coefficients) in the ctilde file format, e.g. in the potential file format with **.ace* or **.yace* extensions from *pair_style pace*. Note that unlike the potential file, the Clebsch-Gordan coefficients in the descriptor file supplied should *NOT* be multiplied by linear or square root embedding terms.

The SNAP descriptor file closely follows the format of the *pair_style snap* parameter file. The file can contain blank and comment lines (start with #) anywhere. Each non-blank non-comment line must contain one keyword/value pair. The required keywords are *rcutfac* and *twojmax*. There are many optional keywords that are described on the *pair_style snap* doc page. In addition, the SNAP descriptor file must contain the *nelems*, *elems*, *radelems*, and *welems* keywords. The *nelems* keyword specifies the number of elements provided in the other three keywords. The *elems* keyword is followed by a list of *nelems* element names that must include the element names appearing in the *pair_coeff* command, but can contain other names too. Similarly, the *radelems* and *welems* keywords are followed by lists of *nelems* numbers

giving the element radius and element weight of each element. Obviously, the order in which the elements are listed must be consistent for all three keywords.

The SO3 descriptor file is similar to the SNAP descriptor except that it contains a few more arguments (e.g., *nmax* and *alpha*). The preparation of SO3 descriptor and model files can be done with the Pyxtal_FF package.

The ACE descriptor file differs from the SNAP and SO3 files. It more closely resembles the potential file format for linear or square-root embedding ACE potentials used in the *pair_style pace*. As noted above, the key difference is that the Clebsch-Gordan coefficients in the descriptor file with *mliap descriptor ace* are *NOT* multiplied by linear or square root embedding terms. In other words,the model is separated from the descriptor definitions and hyperparameters. In *pair_style pace*, they are combined. The ACE descriptor files required by *mliap* are generated automatically in FitSNAP during linear, pytorch, etc. ACE model fitting. Additional tools are provided there to prepare *ace* descriptor files and hyperparameters before model fitting. The *ace* descriptor files can also be extracted from ACE model fits in python-ace.. It is important to note that order of the types listed in *pair_coeff* must match the order of the elements/types listed in the ACE descriptor file for all *mliap* styles when using *ace* descriptors.

See the *pair_coeff* page for alternate ways to specify the path for these *model* and *descriptor* files.

Note

To significantly reduce SO3 descriptor/force calculation time, some properties are pre-computed and reused during the calculation. These can consume a significant amount of RAM for simulations of larger systems since their size depends on the total number of neighbors per MPI process.

Added in version 3Nov2022.

The *unified* keyword is followed by an argument specifying the filename containing the serialized unified Python object and the "ghostneigh" toggle (0/1) to disable/enable the construction of neighbors lists including neighbors of ghost atoms. If the filename ends in '.pt', or '.pth', it will be loaded using pytorch; otherwise, it will be loaded as a pickle file. If ghostneigh is enabled, it is recommended to set *comm_modify* cutoff manually, such as in the following example.

```
variable ninteractions equal 2
variable cutdist equal 7.5
variable skin equal 1.0
variable commcut equal (${ninteractions}*${cutdist})+${skin}
neighbor ${skin} bin
comm_modify cutoff ${commcut}
```

Note

To load a model from memory (i.e. an existing python object), call *lammps.mliap.load_unified(unified)* from python, and then specify the filename as "EXISTS". When using LAMMPS via the library mode, you will need to call *lammps.mliappy.activate_mliappy(lmp)* on the active LAMMPS object before the pair style is defined. This call locates and loads the mliap-specific python module that is built into LAMMPS.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.186.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS with user-specifiable parameters as described above. You never need to specify a pair_coeff command with I != J arguments for this style.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.186.5 Restrictions

This pair style is part of the ML-IAP package. It is only enabled if LAMMPS was built with that package. In addition, building LAMMPS with the ML-IAP package requires building LAMMPS with the ML-SNAP package. The *mliappy* model requires building LAMMPS with the PYTHON package. The *ace* descriptor requires building LAMMPS with the ML-PACE package. See the *Build package* page for more info. Note that *pair_mliap/kk* acceleration will *not* invoke the *kk* accelerated variants of SNAP or ACE descriptors.

4.186.6 Related commands

pair_style snap, *compute mliap*

4.186.7 Default

none

(Bartok2013) Bartok, Kondor, Csanyi, Phys Rev B, 87, 184115 (2013).

(Zagaceta2020) Zagaceta, Yanxon, Zhu, J Appl Phys, 128, 045113 (2020).

(Yanxon2020) Yanxon, Zagaceta, Tang, Matteson, Zhu, Mach. Learn.: Sci. Technol. 2, 027001 (2020).

4.187 pair_style momb command

4.187.1 Syntax

```
pair_style momb cutoff s6 d
```

- cutoff = global cutoff (distance units)
- s6 = global scaling factor of the exchange-correlation functional used (unitless)
- d = damping scaling factor of Grimme's method (unitless)

4.187.2 Examples

```
pair_style momb 12.0 0.75 20.0
pair_style hybrid/overlay eam/fs lj/charmm/coul/long 10.0 12.0 momb 12.0 0.75 20.0 morse 5.5

pair_coeff 1 2 momb 0.0 1.0 1.0 10.2847 2.361
```

4.187.3 Description

Style *momb* computes pairwise van der Waals (vdW) and short-range interactions using the Morse potential and *(Grimme)* method implemented in the Many-Body Metal-Organic (MOMB) force field described comprehensively in *(Fichthorn)* and *(Zhou)*. Grimme's method is widely used to correct for dispersion in density functional theory calculations.

$$E = D_0[\exp^{-2\alpha(r-r_0)} - 2\exp^{-\alpha(r-r_0)}] - s_6\frac{C_6}{r^6}f_{damp}(r,R_r)$$

$$f_{damp}(r,R_r) = \frac{1}{1+\exp^{-d(r/R_r-1)}}$$

For the *momb* pair style, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* as described below:

- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)
- C_6 (energy*distance^6 units)
- R_r (distance units, typically sum of atomic vdW radii)

4.187.4 Restrictions

This style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS is built with that package. See the *Build package* page on for more info.

4.187.5 Related commands

pair_coeff, *pair_style morse*

4.187.6 Default

none

(Grimme) Grimme, J Comput Chem, 27(15), 1787-1799 (2006).

(Fichthorn) Fichthorn, Balankura, Qi, CrystEngComm, 18(29), 5410-5417 (2016).

(Zhou) Zhou, Saidi, Fichthorn, J Phys Chem C, 118(6), 3366-3374 (2014).

4.188 pair_style morse command

Accelerator Variants: *morse/gpu*, *morse/omp*, *morse/opt*, *morse/kk*

4.189 pair_style morse/smooth/linear command

Accelerator Variants: *morse/smooth/linear/omp*

4.189.1 Syntax

```
pair_style style args
```

- style = *morse* or *morse/smooth/linear* or *morse/soft*
- args = list of arguments for a particular style

morse args = cutoff
  cutoff = global cutoff for Morse interactions (distance units)
morse/smooth/linear args = cutoff
  cutoff = global cutoff for Morse interactions (distance units)

4.189.2 Examples

```
pair_style morse 2.5
pair_style morse/smooth/linear 2.5
pair_coeff * * 100.0 2.0 1.5
pair_coeff 1 1 100.0 2.0 1.5 3.0
```

4.189.3 Description

Style *morse* computes pairwise interactions with the formula

$$E = D_0 \left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)} \right] \qquad r < r_c$$

r_c is the cutoff.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global morse cutoff is used.

The *morse/smooth/linear* variant is similar to the lj/smooth/linear variant in that it adds to the potential a shift and a linear term so that both, potential energy and force, go to zero at the cut-off:

$$\phi(r) = D_0 \left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)} \right] \qquad r < r_c$$

$$E(r) = \phi(r) - \phi(r_c) - (r - r_c) \left. \frac{d\phi}{dr} \right|_{r=r_c} \qquad r < r_c$$

The syntax of the pair_style and pair_coeff commands are the same for the *morse* and *morse/smooth/linear* styles.

A version of the *morse* style with a soft core, *morse/soft*, suitable for use in free energy calculations, is part of the FEP package and is documented with the *pair_style */soft* styles. The version with soft core is only available if LAMMPS was built with that package. See the *Build package* page for more info.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.189.4 Mixing, shift, table, tail correction, restart, rRESPA info

None of these pair styles support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

All of these pair styles support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table options are not relevant for the Morse pair styles.

None of these pair styles support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

All of these pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.189.5 Restrictions

The *morse/smooth/linear* pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

4.189.6 Related commands

pair_coeff, *pair_style */soft*

4.189.7 Default

none

4.190 pair_style multi/lucy command

4.190.1 Syntax

```
pair_style multi/lucy style N keyword ...
```

- style = *lookup* or *linear* = method of interpolation
- N = use N values in *lookup*, *linear* tables

4.190.2 Examples

```
pair_style multi/lucy linear 1000
pair_coeff * * multibody.table ENTRY1 7.0
```

4.190.3 Description

Style *multi/lucy* computes a density-dependent force following from the many-body form described in *(Moore)* and *(Warren)* as

$$F_i^{DD}(\rho_i, \rho_j, r_{ij}) = \frac{1}{2}\omega_{DD}\left(r_{ij}\right)\left[A\left(\rho_i\right) + A\left(\rho_j\right)\right]e_{ij}$$

which consists of a density-dependent function, $A(\rho)$, and a radial-dependent weight function, $\omega_{DD}(r_{ij})$. The radial-dependent weight function, $\omega_{DD}(r_{ij})$, is taken as the Lucy function:

$$\omega_{DD}\left(r_{ij}\right) = \left(1 + \frac{3r_{ij}}{r_{cut}}\right)\left(1 + \frac{r_{ij}}{r_{cut}}\right)^3$$

The density-dependent energy for a given particle is given by:

$$u_i^{DD}\left(\rho_i\right) = \frac{\pi r_{cut}^4}{84}\int_{\rho_0}^{\rho_i} A\left(\rho'\right)d\rho'$$

See the supporting information of *(Brennan)* or the publication by *(Moore)* for more details on the functional form.

An interpolation table is used to evaluate the density-dependent energy ($\int A(\rho')d\rho'$) and force ($A(\rho')$). Note that the prefactor to the energy is computed after the interpolation, thus the $\int A(\rho')d\rho'$ will have units of energy / length^4.

The interpolation table is created as a pre-computation by fitting cubic splines to the file values and interpolating the density-dependent energy and force at each of N densities. During a simulation, the tables are used to interpolate the density-dependent energy and force as needed for each pair of particles separated by a distance R. The interpolation is done in one of 2 styles: *lookup* and *linear*.

For the *lookup* style, the density is used to find the nearest table entry, which is the density-dependent energy and force.

For the *linear* style, the density is used to find the 2 surrounding table values from which the density-dependent energy and force are computed by linear interpolation.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- filename
- keyword
- cutoff (distance units)

The filename specifies a file containing the tabulated density-dependent energy and force. The keyword specifies a section of the file. The cutoff is an optional coefficient. If not specified, the outer cutoff in the table itself (see below) will be used to build an interpolation table that extend to the largest tabulated distance. If specified, only file values up to the cutoff are used to create the interpolation table. The format of this file is described below.

The format of a tabulated file is a series of one or more sections, defined as follows (without the parenthesized comments):

```
# Density-dependent function (one or more comment or blank lines)

DD-FUNCTION                 (keyword is first text on line)
N 500 R 1.0 10.0          (N, R, RSQ parameters)
                        (blank)
1 1.0 25.5 102.34         (index, density, energy/r^4, force)
2 1.02 23.4 98.5
...
500 10.0 0.001 0.003
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The line can contain additional text, but the initial text must match the argument specified in the pair_coeff command. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required and its value is the number of table entries that follow. Note that this may be different than the *N* specified in the *pair_style multi/lucy* command. Let Ntable = *N* in the pair_style command, and Nfile = "N" in the tabulated file. What LAMMPS does is a preliminary interpolation by creating splines using the Nfile tabulated values as nodal points. It uses these to interpolate the density-dependent energy and force at Ntable different points. The resulting tables of length Ntable are then used as described above, when computing the density-dependent energy and force. This means that if you want the interpolation tables of length Ntable to match exactly what is in the tabulated file (with effectively no preliminary interpolation), you should set Ntable = Nfile, and use the "RSQ" parameter. This is because the internal table abscissa is always RSQ (separation distance squared), for efficient lookup.

All other parameters are optional. If "R" or "RSQ" does not appear, then the distances in each line of the table are used as-is to perform spline interpolation. In this case, the table values can be spaced in *density* uniformly or however you wish to position table values in regions of large gradients.

If used, the parameters "R" or "RSQ" are followed by 2 values *rlo* and *rhi*. If specified, the density associated with each density-dependent energy and force value is computed from these 2 values (at high accuracy), rather than using the (low-accuracy) value listed in each line of the table. The density values in the table file are ignored in this case. For "R", distances uniformly spaced between *rlo* and *rhi* are computed; for "RSQ", squared distances uniformly spaced between *rlo*rlo* and *rhi*rhi* are computed.

Note

If you use "R" or "RSQ", the tabulated distance values in the file are effectively ignored, and replaced by new values as described in the previous paragraph. If the density value in the table is not very close to the new value (i.e. round-off difference), then you will be assigning density-dependent energy and force values to a different density, which is probably not what you want. LAMMPS will warn if this is occurring.

Following a blank line, the next N lines list the tabulated values. On each line, the first value is the index from 1 to N, the second value is r (in density units), the third value is the density-dependent function value (in energy units / length^4), and the fourth is the force (in force units). The density values must increase from one line to the next.

Note that one file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified keyword.

4.190.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The *pair_modify* shift, table, and tail options are not relevant for this pair style.

This pair style writes the settings for the "pair_style multi/lucy" command to *binary restart files*, so a pair_style command does not need to specified in an input script that reads a restart file. However, the coefficient information is not stored in the restart file, since it is tabulated in the potential files. Thus, pair_coeff commands do need to be specified in the restart input script.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.190.5 Restrictions

This command is part of the DPD-REACT package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.190.6 Related commands

pair_coeff

4.190.7 Default

none

(Warren) Warren, Phys Rev E, 68, 066702 (2003).

(Brennan) Brennan, J Chem Phys Lett, 5, 2144-2149 (2014).

(Moore) Moore, J Chem Phys, 144, 104501 (2016).

4.191 pair_style multi/lucy/rx command

Accelerator Variants: *multi/lucy/rx/kk*

4.191.1 Syntax

```
pair_style multi/lucy/rx style N keyword ...
```

- style = *lookup* or *linear* = method of interpolation
- N = use N values in *lookup*, *linear* tables
- weighting = fractional or molecular (optional)

4.191.2 Examples

```
pair_style multi/lucy/rx linear 1000
pair_style multi/lucy/rx linear 1000 fractional
pair_style multi/lucy/rx linear 1000 molecular
pair_coeff * * multibody.table ENTRY1 h2o h2o 7.0
pair_coeff * * multibody.table ENTRY1 h2o 1fluid 7.0
```

4.191.3 Description

Style *multi/lucy/rx* is used in reaction DPD simulations, where the coarse-grained (CG) particles are composed of *m* species whose reaction rate kinetics are determined from a set of *n* reaction rate equations through the *fix rx* command. The species of one CG particle can interact with a species in a neighboring CG particle through a site-site interaction potential model. Style *multi/lucy/rx* computes the site-site density-dependent force following from the many-body form described in *(Moore)* and *(Warren)* as

$$F_i^{DD}(\rho_i, \rho_j, r_{ij}) = \frac{1}{2}\omega_{DD}\left(r_{ij}\right)\left[A\left(\rho_i\right) + A\left(\rho_j\right)\right]e_{ij}$$

which consists of a density-dependent function, $A(\rho)$, and a radial-dependent weight function, $\omega_{DD}(r_{ij})$. The radial-dependent weight function, $\omega_{DD}(r_{ij})$, is taken as the Lucy function:

$$\omega_{DD}\left(r_{ij}\right) = \left(1 + \frac{3r_{ij}}{r_{cut}}\right)\left(1 + \frac{r_{ij}}{r_{cut}}\right)^3$$

The density-dependent energy for a given particle is given by:

$$u_i^{DD}\left(\rho_i\right) = \frac{\pi r_{cut}^4}{84}\int_{\rho_0}^{\rho_i} A\left(\rho'\right)d\rho'$$

See the supporting information of *(Brennan)* or the publication by *(Moore)* for more details on the functional form.

An interpolation table is used to evaluate the density-dependent energy ($\int A(\rho')d\rho'$) and force ($A(\rho')$). Note that the prefactor to the energy is computed after the interpolation, thus the $\int A(\rho')d\rho'$ will have units of energy / length^4.

The interpolation table is created as a pre-computation by fitting cubic splines to the file values and interpolating the density-dependent energy and force at each of N densities. During a simulation, the tables are used to interpolate the density-dependent energy and force as needed for each pair of particles separated by a distance R. The interpolation is done in one of 2 styles: *lookup* and *linear*.

For the *lookup* style, the density is used to find the nearest table entry, which is the density-dependent energy and force.

For the *linear* style, the density is used to find the 2 surrounding table values from which the density-dependent energy and force are computed by linear interpolation.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- filename
- keyword
- species1
- species2
- cutoff (distance units)

The filename specifies a file containing the tabulated density-dependent energy and force. The keyword specifies a section of the file. The cutoff is an optional coefficient. If not specified, the outer cutoff in the table itself (see below) will be used to build an interpolation table that extend to the largest tabulated distance. If specified, only file values up to the cutoff are used to create the interpolation table. The format of this file is described below.

The species tags define the site-site interaction potential between two species contained within two different particles. The species tags must either correspond to the species defined in the reaction kinetics files specified with the *fix rx* command or they must correspond to the tag "1fluid", signifying interaction with a product species mixture determined through a one-fluid approximation. The interaction potential is weighted by the geometric average of either the mole fraction concentrations or the number of molecules associated with the interacting coarse-grained particles (see the *fractional* or *molecular* weighting pair style options). The coarse-grained potential is stored before and after the reaction kinetics solver is applied, where the difference is defined to be the internal chemical energy (uChem).

The format of a tabulated file is a series of one or more sections, defined as follows (without the parenthesized comments):

```
# Density-dependent function (one or more comment or blank lines)

DD-FUNCTION                (keyword is first text on line)
N 500 R 1.0 10.0           (N, R, RSQ parameters)
                           (blank)
1 1.0 25.5 102.34          (index, density, energy/r^4, force)
2 1.02 23.4 98.5
...
500 10.0 0.001 0.003
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The line can contain additional text, but the initial text must match the argument specified in the pair_coeff command. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required and its value is the number of table entries that follow. Note that this may be different than the *N* specified in the *pair_style multi/lucy/rx* command. Let Ntable = *N* in the pair_style command, and Nfile = "N" in the tabulated file. What LAMMPS does is a preliminary interpolation by creating splines using the Nfile tabulated values as nodal points. It uses these to interpolate the density-dependent energy and force at Ntable different points. The resulting tables of length Ntable are then used as described above, when computing the density-dependent energy and force. This means that if you want the interpolation tables of length Ntable to match exactly what is in the tabulated file (with effectively no preliminary interpolation), you should set Ntable = Nfile, and use the "RSQ" parameter. This is because the internal table abscissa is always RSQ (separation distance squared), for efficient lookup.

All other parameters are optional. If "R" or "RSQ" does not appear, then the distances in each line of the table are used as-is to perform spline interpolation. In this case, the table values can be spaced in *density* uniformly or however you wish to position table values in regions of large gradients.

If used, the parameters "R" or "RSQ" are followed by 2 values *rlo* and *rhi*. If specified, the density associated with each density-dependent energy and force value is computed from these 2 values (at high accuracy), rather than using the (low-accuracy) value listed in each line of the table. The density values in the table file are ignored in this case. For "R", distances uniformly spaced between *rlo* and *rhi* are computed; for "RSQ", squared distances uniformly spaced between *rlo***rlo* and *rhi***rhi* are computed.

Note

If you use "R" or "RSQ", the tabulated distance values in the file are effectively ignored, and replaced by new values as described in the previous paragraph. If the density value in the table is not very close to the new value (i.e. round-off difference), then you will be assigning density-dependent energy and force values to a different density, which is probably not what you want. LAMMPS will warn if this is occurring.

Following a blank line, the next N lines list the tabulated values. On each line, the first value is the index from 1 to N, the second value is r (in density units), the third value is the density-dependent function value (in energy units / length^4), and the fourth is the force (in force units). The density values must increase from one line to the next.

Note that one file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified keyword.

4.191.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The *pair_modify* shift, table, and tail options are not relevant for this pair style.

This pair style writes the settings for the "pair_style multi/lucy/rx" command to *binary restart files*, so a pair_style command does not need to specified in an input script that reads a restart file. However, the coefficient information is not stored in the restart file, since it is tabulated in the potential files. Thus, pair_coeff commands do need to be specified in the restart input script.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.191.5 Restrictions

This command is part of the DPD-REACT package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.191.6 Related commands

pair_coeff

4.191.7 Default

fractional weighting

(Warren) Warren, Phys Rev E, 68, 066702 (2003).

(Brennan) Brennan, J Chem Phys Lett, 5, 2144-2149 (2014).

(Moore) Moore, J Chem Phys, 144, 104501 (2016).

4.192 pair_style nb3b/harmonic command

4.193 pair_style nb3b/screened command

4.193.1 Syntax

```
pair_style style
```

- style = *nb3b/harmonic* or *nb3b/screened*

4.193.2 Examples

```
pair_style nb3b/harmonic
pair_coeff * * MgOH.nb3bharmonic Mg O H

pair_style nb3b/screened
pair_coeff * * PO.nb3b.screened P NULL O
pair_coeff * * SiOH.nb3b.screened Si O H
```

4.193.3 Description

The pair style *nb3b/harmonic* computes a non-bonded 3-body harmonic potential for the energy E of a system of atoms as

$$E = K(\theta - \theta_0)^2$$

where θ_0 is the equilibrium value of the angle and K is a prefactor. Note that the usual 1/2 factor is included in K. The form of the potential is identical to that used in angle_style *harmonic*, but in this case, the atoms do not need to be explicitly bonded.

Style *nb3b/screened* adds an additional exponentially decaying factor to the harmonic term, given by

$$E = K(\theta - \theta_0)^2 \exp\left(-\frac{r_{ij}}{\rho_{ij}} - \frac{r_{ik}}{\rho_{ik}}\right)$$

where $\rho_i j$ and $\rho_i k$ are the screening factors along the two bonds. Note that the usual 1/2 factor is included in K.

Only a single pair_coeff command is used with these styles which specifies a potential file with parameters for specified elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.nb3b.harmonic has potential values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.nb3b.harmonic Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the potential file. The final C argument maps LAMMPS atom type 4 to the C element in the potential file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when the potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials. Two examples of pair_coeff command for use with the *hybrid* pair style are:

```
pair_coeff * * nb3b/harmonic MgOH.nb3b.harmonic Mg O H
```

Three-body non-bonded harmonic files in the *potentials* directory of the LAMMPS distribution have a ".nb3b.harmonic" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements.

Each entry has six arguments. The first three are atom types as referenced in the LAMMPS input file. The first argument specifies the central atom. The fourth argument indicates the K parameter. The fifth argument indicates θ_0. The sixth argument indicates a separation cutoff in Angstroms.

For a given entry, if the second and third arguments are identical, then the entry is for a cutoff for the distance between types 1 and 2 (values for K and θ_0 are irrelevant in this case).

For a given entry, if the first three arguments are all different, then the entry is for the K and θ_0 parameters (the cutoff in this case is irrelevant).

It is required that the potential file contains entries for *all* permutations of the elements listed in the pair_coeff command. If certain combinations are not parameterized the corresponding parameters should be set to zero. The potential file can also contain entries for additional elements which are not used in a particular simulation; LAMMPS ignores those entries.

4.193.4 Restrictions

This pair style can only be used if LAMMPS was built with the MANYBODY package. See the *Build package* page for more info.

4.193.5 Related commands

pair_coeff

4.193.6 Default

none

4.194 pair_style nm/cut command

Accelerator Variants: *nm/cut/omp*

4.195 pair_style nm/cut/split command

4.196 pair_style nm/cut/coul/cut command

Accelerator Variants: *nm/cut/coul/cut/omp*

4.197 pair_style nm/cut/coul/long command

Accelerator Variants: *nm/cut/coul/long/omp*

4.197.1 Syntax

```
pair_style style args
```

- style = *nm/cut* or *nm/cut/split* or *nm/cut/coul/cut* or *nm/cut/coul/long*
- args = list of arguments for a particular style

  nm/cut args = cutoff
    cutoff = global cutoff for Pair interactions (distance units)
  nm/cut/split args = cutoff
    cutoff = global cutoff for Pair interactions (distance units)
  nm/cut/coul/cut args = cutoff (cutoff2)
    cutoff = global cutoff for Pair (and Coulombic if only 1 arg) (distance units)
    cutoff2 = global cutoff for Coulombic (optional) (distance units)
  nm/cut/coul/long args = cutoff (cutoff2)
    cutoff = global cutoff for Pair (and Coulombic if only 1 arg) (distance units)
    cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.197.2 Examples

```
pair_style nm/cut 12.0
pair_coeff * * 0.01 5.4 8.0 7.0
pair_coeff 1 1 0.01 4.4 7.0 6.0

pair_style nm/cut/split 1.12246
pair_coeff 1 1 1.0 1.1246 12 6
pair_coeff * * 1.0 1.1246 11 6

pair_style nm/cut/coul/cut 12.0 15.0
pair_coeff * * 0.01 5.4 8.0 7.0
pair_coeff 1 1 0.01 4.4 7.0 6.0

pair_style nm/cut/coul/long 12.0 15.0
pair_coeff * * 0.01 5.4 8.0 7.0
pair_coeff 1 1 0.01 4.4 7.0 6.0
```

4.197.3 Description

Style *nm* computes site-site interactions based on the N-M potential by *Clarke*, mainly used for ionic liquids. A site can represent a single atom or a united-atom site. The energy of an interaction has the following form:

$$E = \frac{E_0}{(n-m)} \left[m \left(\frac{r_0}{r} \right)^n - n \left(\frac{r_0}{r} \right)^m \right] \qquad r < r_c$$

where r_c is the cutoff and r_0 is the minimum of the potential. Please note that this differs from the convention used for other Lennard-Jones potentials in LAMMPS where σ represents the location where the energy is zero.

Style *nm/cut/split* applies the standard LJ (12-6) potential above $r_0 = 2^{\frac{1}{6}}\sigma$. Style *nm/cut/split* is employed in polymer equilibration protocols that combine core-softening approaches with topology-changing moves *Dietz*.

Style *nm/cut/coul/cut* adds a Coulombic pairwise interaction given by

$$E = \frac{C q_i q_j}{\varepsilon r} \qquad r < r_c$$

where C is an energy-conversion constant, q_i and q_j are the charges on the two atoms, and epsilon is the dielectric constant which can be set by the *dielectric* command. If one cutoff is specified in the pair_style command, it is used for both the N-M and Coulombic terms. If two cutoffs are specified, they are used as cutoffs for the N-M and Coulombic terms respectively.

Styles *nm/cut/coul/long* compute the same Coulombic interactions as style *nm/cut/coul/cut* except that an additional damping factor is applied to the Coulombic term so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

For all of the *nm* pair styles, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands.

- E_0 (energy units)
- r_0 (distance units)
- n (unitless)
- m (unitless)
- cutoff1 (distance units)
- cutoff2 (distance units)

The latter 2 coefficients are optional. If not specified, the global N-M and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both N-M and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the N-M and Coulombic cutoffs for this type pair. You cannot specify 2 cutoffs for style *nm*, since it has no Coulombic terms.

For *nm/cut/coul/long* only the N-M cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

4.197.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

All of the *nm* pair styles supports the *pair_modify* shift option for the energy of the pair interaction.

The *nm/cut/coul/long* pair styles support the *pair_modify* table option since they can tabulate the short-range portion of the long-range Coulombic interaction.

All of the *nm* pair styles support the *pair_modify* tail option for adding a long-range tail correction to the energy and pressure for the N-M portion of the pair interaction.

All of the *nm* pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

All of the *nm* pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.197.5 Restrictions

These pair styles are part of the EXTRA-PAIR package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.197.6 Related commands

pair_coeff, *pair style lj/cut*, *bond style fene/nm*

4.197.7 Default

none

(Clarke) Clarke and Smith, J Chem Phys, 84, 2290 (1986).

(Dietz) Dietz and Hoy, J. Chem Phys, 156, 014103 (2022).

4.198 pair_style none command

4.198.1 Syntax

```
pair_style none
```

4.198.2 Examples

```
pair_style none
```

4.198.3 Description

Using a pair style of *none* means that any previous pair style setting will be deleted and pairwise forces and energies are not computed.

As a consequence there will be a pairwise force cutoff of 0.0, which has implications for the default setting of the neighbor list and the communication cutoff. Those are the sum of the largest pairwise cutoff and the neighbor skin distance (see the documentation of the *neighbor* command and the *comm_modify* command). When you have bonds, angles, dihedrals, or impropers defined at the same time, you must set the communication cutoff so that communication cutoff distance is large enough to acquire and communicate sufficient ghost atoms from neighboring subdomains as needed for computing bonds, angles, etc.

A pair style of *none* will also not request a pairwise neighbor list. However if the *neighbor* style is *bin*, data structures for binning are still allocated. If the neighbor list cutoff is small, then these data structures can consume a large amount of memory. So you should either set the neighbor style to *nsq* or set the skin distance to a larger value.

See the *pair_style zero* for a way to set a pairwise cutoff and thus trigger the building of a neighbor lists and setting a corresponding communication cutoff, but compute no pairwise interactions.

4.198.4 Restrictions

You must not use a *pair_coeff* command with this pair style. Since there is no interaction computed, you cannot set any coefficients for it.

4.198.5 Related commands

pair_style zero

4.198.6 Default

none

4.199 pair_style oxdna/excv command

4.200 pair_style oxdna/stk command

4.201 pair_style oxdna/hbond command

4.202 pair_style oxdna/xstk command

4.203 pair_style oxdna/coaxstk command

4.203.1 Syntax

```
pair_style style1

pair_coeff * * style2 args
```

- style1 = *hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk*
- style2 = *oxdna/excv* or *oxdna/stk* or *oxdna/hbond* or *oxdna/xstk* or *oxdna/coaxstk*
- args = list of arguments for these particular styles

oxdna/stk args = seq T xi kappa 6.0 0.4 0.9 0.32 0.75 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 2.0 0.65 2.0 0.65
  seq = seqav (for average sequence stacking strength) or seqdep (for sequence-dependent stacking strength)
  T = temperature (LJ units: 0.1 = 300 K, real units: 300 = 300 K)
  xi = 1.3448 (LJ units) or 8.01727944817084 (real units), temperature-independent coefficient in stacking strength
  kappa = 2.6568 (LJ units) or 0.005279604 (real units), coefficient of linear temperature dependence in stacking strength
oxdna/hbond args = seq eps 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
  seq = seqav (for average sequence base-pairing strength) or seqdep (for sequence-dependent base-pairing strength)
  eps = 1.077 (LJ units) or 6.42073911784652 (real units), average hydrogen bonding strength between A-T and C-G Watson-Crick base pairs, 0 between all other pairs

4.203.2 Examples

```
# LJ units
pair_style hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk
pair_coeff * * oxdna/excv    2.0 0.7 0.675 2.0 0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxdna/stk     seqdep 0.1 1.3448 2.6568 6.0 0.4 0.9 0.32 0.75 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 2.0 0.65 2.0 0.65
pair_coeff * * oxdna/hbond   seqdep 0.0 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 1 4 oxdna/hbond   seqdep 1.077 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 2 3 oxdna/hbond   seqdep 1.077 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.
```

(continues on next page)

(continued from previous page)

```
↪141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff * * oxdna/xstk    47.5 0.575 0.675 0.495 0.655 2.25 0.791592653589793 0.58 1.7 1.0 0.68 1.7 1.0␣
↪0.68 1.5 0 0.65 1.7 0.875 0.68 1.7 0.875 0.68
pair_coeff * * oxdna/coaxstk 46.0 0.4 0.6 0.22 0.58 2.0 2.541592653589793 0.65 1.3 0 0.8 0.9 0 0.95 0.9 0␣
↪0.95 2.0 -0.65 2.0 -0.65

pair_style hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk
pair_coeff * * oxdna/excv    oxdna_lj.cgdna
pair_coeff * * oxdna/stk     seqav 0.1 1.3448 2.6568 oxdna_lj.cgdna
pair_coeff * * oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff 1 4 oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff 2 3 oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff * * oxdna/xstk    oxdna_lj.cgdna
pair_coeff * * oxdna/coaxstk oxdna_lj.cgdna

# Real units
pair_style hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk
pair_coeff * * oxdna/excv    11.92337812042065 5.9626 5.74965 11.92337812042065 4.38677 4.259 11.
↪92337812042065 2.81094 2.72576
pair_coeff * * oxdna/stk     seqdep 300.0 8.01727944817084 0.005279604 0.70439070204273 3.4072 7.6662␣
↪2.72576 6.3885 1.3 0.0 0.8 0.9 0.0 0.95 0.9 0.0 0.95 2.0 0.65 2.0 0.65
pair_coeff * * oxdna/hbond   seqdep 0.0 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.5 0.0 0.7 1.5 0.
↪0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff 1 4 oxdna/hbond   seqdep 6.42073911784652 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.
↪5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff 2 3 oxdna/hbond   seqdep 6.42073911784652 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.
↪5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff * * oxdna/xstk    3.9029021145006 4.89785 5.74965 4.21641 5.57929 2.25 0.791592654 0.58 1.7␣
↪1.0 0.68 1.7 1.0 0.68 1.5 0.0 0.65 1.7 0.875 0.68 1.7 0.875 0.68
pair_coeff * * oxdna/coaxstk 3.77965257404268 3.4072 5.1108 1.87396 4.94044 2.0 2.541592654 0.65 1.3 0.
↪0 0.8 0.9 0.0 0.95 0.9 0.0 0.95 2.0 -0.65 2.0 -0.65

pair_style hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk
pair_coeff * * oxdna/excv    oxdna_real.cgdna
pair_coeff * * oxdna/stk     seqav 300.0 8.01727944817084 0.005279604 oxdna_real.cgdna
pair_coeff * * oxdna/hbond   seqav oxdna_real.cgdna
pair_coeff 1 4 oxdna/hbond   seqav oxdna_real.cgdna
pair_coeff 2 3 oxdna/hbond   seqav oxdna_real.cgdna
pair_coeff * * oxdna/xstk    oxdna_real.cgdna
pair_coeff * * oxdna/coaxstk oxdna_real.cgdna
```

Note

The coefficients in the above examples are provided in forms compatible with both *units lj* and *units real* (see documentation of *units*). These can also be read from a potential file with correct unit style by specifying the name of the file. Several potential files for each unit style are included in the potentials directory of the LAMMPS distribution.

4.203.3 Description

The *oxdna* pair styles compute the pairwise-additive parts of the oxDNA force field for coarse-grained modelling of DNA. The effective interaction between the nucleotides consists of potentials for the excluded volume interaction *oxdna/excv*, the stacking *oxdna/stk*, cross-stacking *oxdna/xstk* and coaxial stacking interaction *oxdna/coaxstk* as well as the hydrogen-bonding interaction *oxdna/hbond* between complementary pairs of nucleotides on opposite strands. Average sequence or sequence-dependent stacking and base-pairing strengths are supported *(Sulc)*. Quasi-unique base-pairing between nucleotides can be achieved by using more complementary pairs of atom types like 5-8 and 6-7, 9-12 and 10-11, 13-16 and 14-15, etc. This prevents the hybridization of in principle complementary bases within Ntypes/4 bases up and down along the backbone.

The exact functional form of the pair styles is rather complex. The individual potentials consist of products of modulation factors, which themselves are constructed from a number of more basic potentials (Morse, Lennard-Jones, harmonic angle and distance) as well as quadratic smoothing and modulation terms. We refer to *(Ouldridge-DPhil)* and *(Ouldridge)* for a detailed description of the oxDNA force field.

Note

These pair styles have to be used together with the related oxDNA bond style *oxdna/fene* for the connectivity of the phosphate backbone (see also documentation of *bond_style oxdna/fene*). Most of the coefficients in the above example have to be kept fixed and cannot be changed without reparameterizing the entire model. Exceptions are the first four coefficients after *oxdna/stk* (seq=seqdep, T=0.1, xi=1.3448 and kappa=2.6568 and corresponding *real unit* equivalents in the above examples) and the first coefficient after *oxdna/hbond* (seq=seqdep in the above example). When using a Langevin thermostat, e.g. through *fix langevin* or *fix nve/dotc/langevin* the temperature coefficients have to be matched to the one used in the fix.

Note

These pair styles have to be used with the *atom_style hybrid bond ellipsoid oxdna* (see documentation of *atom_style*). The *atom_style oxdna* stores the 3'-to-5' polarity of the nucleotide strand, which is set through the bond topology in the data file. The first (second) atom in a bond definition is understood to point towards the 3'-end (5'-end) of the strand.

Example input and data files for DNA duplexes can be found in examples/PACKAGES/cgdna/examples/oxDNA/ and .../oxDNA2/. A simple python setup tool which creates single straight or helical DNA strands, DNA duplexes or arrays of DNA duplexes can be found in examples/PACKAGES/cgdna/util/.

Please cite *(Henrich)* in any publication that uses this implementation. An updated documentation that contains general information on the model, its implementation and performance as well as the structure of the data and input file can be found here.

Please cite also the relevant oxDNA publications *(Ouldridge)*, *(Ouldridge-DPhil)* and *(Sulc)*.

4.203.4 Potential file reading

For each pair style above the first non-modifiable argument can be a filename, and if it is, no further arguments should be supplied. Therefore the following command:

```
pair_coeff 1 4 oxdna/hbond   seqav oxdna_lj.cgdna
```

will be interpreted as a request to read the corresponding hydrogen bonding potential parameters from the file with the given name. The file can define multiple potential parameters for both bonded and pair interactions, but for the example pair interaction above there must exist in the file a line of the form:

```
1 4 hbond     <coefficients>
```

If potential customization is required, the potential file reading can be mixed with the manual specification of the potential parameters. For example, the following command:

```
pair_style hybrid/overlay oxdna/excv oxdna/stk oxdna/hbond oxdna/xstk oxdna/coaxstk
pair_coeff * * oxdna/excv    oxdna_lj.cgdna
pair_coeff * * oxdna/stk     seqav 0.1 1.3448 2.6568 6.0 0.4 0.9 0.32 0.75 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 2.
↪0 0.65 2.0 0.65
pair_coeff * * oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff 1 4 oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff 2 3 oxdna/hbond   seqav oxdna_lj.cgdna
pair_coeff * * oxdna/xstk    oxdna_lj.cgdna
pair_coeff * * oxdna/coaxstk 46.0 0.4 0.6 0.22 0.58 2.0 2.541592653589793 0.65 1.3 0 0.8 0.9 0 0.95 0.9 0␣
↪0.95 2.0 -0.65 2.0 -0.65
```

will read the stacking and coaxial stacking potential parameters from the manual specification and all others from the potential file *oxdna_lj.cgdna*.

There are sample potential files for each unit style in the potentials directory of the LAMMPS distribution. The potential file unit system must align with the units defined via the *units* command. For conversion between different *LJ* and *real* unit systems for oxDNA, the python tool *lj2real.py* located in the examples/PACKAGES/cgdna/util/ directory can be used. This tool assumes similar file structure to the examples found in examples/PACKAGES/cgdna/examples/.

4.203.5 Restrictions

These pair styles can only be used if LAMMPS was built with the CG-DNA package and the MOLECULE and ASPHERE package. See the *Build package* page for more info.

4.203.6 Related commands

bond_style oxdna/fene, *pair_coeff*, *bond_style oxdna2/fene*, *pair_style oxdna2/excv*, *bond_style oxrna2/fene*, *pair_style oxrna2/excv*, *atom_style oxdna*, *fix nve/dotc/langevin*

4.203.7 Default

none

(Henrich) O. Henrich, Y. A. Gutierrez-Fosado, T. Curk, T. E. Ouldridge, Eur. Phys. J. E 41, 57 (2018).

(Ouldridge-DPhil) T.E. Ouldridge, Coarse-grained modelling of DNA and DNA self-assembly, DPhil. University of Oxford (2011).

(Ouldridge) T.E. Ouldridge, A.A. Louis, J.P.K. Doye, J. Chem. Phys. 134, 085101 (2011).

(Sulc) P. Sulc, F. Romano, T.E. Ouldridge, L. Rovigatti, J.P.K. Doye, A.A. Louis, J. Chem. Phys. 137, 135101 (2012).

4.204 pair_style oxdna2/excv command

4.205 pair_style oxdna2/stk command

4.206 pair_style oxdna2/hbond command

4.207 pair_style oxdna2/xstk command

4.208 pair_style oxdna2/coaxstk command

4.209 pair_style oxdna2/dh command

4.209.1 Syntax

```
pair_style style1

pair_coeff * * style2 args
```

- style1 = *hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/dh*
- style2 = *oxdna2/excv* or *oxdna2/stk* or *oxdna2/hbond* or *oxdna2/xstk* or *oxdna2/coaxstk* or *oxdna2/dh*
- args = list of arguments for these particular styles

```
oxdna2/stk args = seq T xi kappa 6.0 0.4 0.9 0.32 0.75 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 2.0 0.65 2.0 0.65
  seq = seqav (for average sequence stacking strength) or seqdep (for sequence-dependent stacking␣
↪strength)
  T = temperature (LJ units: 0.1 = 300 K, real units: 300 = 300 K)
  xi = 1.3523 (LJ units) or 8.06199211612242 (real units), temperature-independent coefficient in stacking␣
↪strength
  kappa = 2.6717 (LJ units) or 0.005309213 (real units), coefficient of linear temperature dependence in␣
↪stacking strength
oxdna2/hbond args = seq eps 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.
↪7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
  seq = seqav (for average sequence base-pairing strength) or seqdep (for sequence-dependent base-pairing␣
↪strength)
```

eps = 1.0678 (LJ units) or 6.36589157849259 (real units), average hydrogen bonding strength between A-T and C-G Watson-Crick base pairs, 0 between all other pairs
oxdna2/dh args = T rhos qeff
T = temperature (LJ units: 0.1 = 300 K, real units: 300 = 300 K)
rhos = salt concentration (mole per litre)
qeff = 0.815 (effective charge in elementary charges)

4.209.2 Examples

```
# LJ units
pair_style hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/dh
pair_coeff * * oxdna2/excv    2.0 0.7 0.675 2.0 0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxdna2/stk     seqdep 0.1 1.3523 2.6717 6.0 0.4 0.9 0.32 0.75 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 2.0 0.65 2.0 0.65
pair_coeff * * oxdna2/hbond   seqdep 0.0 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 1 4 oxdna2/hbond   seqdep 1.0678 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 2 3 oxdna2/hbond   seqdep 1.0678 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff * * oxdna2/xstk    47.5 0.575 0.675 0.495 0.655 2.25 0.791592653589793 0.58 1.7 1.0 0.68 1.7 1.0 0.68 1.5 0 0.65 1.7 0.875 0.68 1.7 0.875 0.68
pair_coeff * * oxdna2/coaxstk 58.5 0.4 0.6 0.22 0.58 2.0 2.891592653589793 0.65 1.3 0 0.8 0.9 0 0.95 0.9 0 0.95 40.0 3.116592653589793
pair_coeff * * oxdna2/dh      0.1 0.5 0.815

pair_style hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/dh
pair_coeff * * oxdna2/excv    oxdna2_lj.cgdna
pair_coeff * * oxdna2/stk     seqdep 0.1 1.3523 2.6717 oxdna2_lj.cgdna
pair_coeff * * oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff 1 4 oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff 2 3 oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff * * oxdna2/xstk    oxdna2_lj.cgdna
pair_coeff * * oxdna2/coaxstk oxdna2_lj.cgdna
pair_coeff * * oxdna2/dh      0.1 0.5 oxdna2_lj.cgdna

# Real units
pair_style hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/dh
pair_coeff * * oxdna2/excv    11.92337812042065 5.9626 5.74965 11.92337812042065 4.38677 4.259 11.92337812042065 2.81094 2.72576
pair_coeff * * oxdna2/stk     seqdep 300.0 8.06199211612242 0.005309213 0.70439070204273 3.4072 7.6662 2.72576 6.3885 1.3 0.0 0.8 0.9 0.0 0.95 0.9 0.0 0.95 2.0 0.65 2.0 0.65
pair_coeff * * oxdna2/hbond   seqdep 0.0 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff 1 4 oxdna2/hbond   seqdep 6.36589157849259 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff 2 3 oxdna2/hbond   seqdep 6.36589157849259 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592654 0.7 4.0 1.570796327 0.45 4.0 1.570796327 0.45
pair_coeff * * oxdna2/xstk    3.9029021145006 4.89785 5.74965 4.21641 5.57929 2.25 0.791592654 0.58 1.
```

(continues on next page)

(continued from previous page)

```
↪7 1.0 0.68 1.7 1.0 0.68 1.5 0.0 0.65 1.7 0.875 0.68 1.7 0.875 0.68
pair_coeff * * oxdna2/coaxstk 4.80673207785863 3.4072 5.1108 1.87396 4.94044 2.0 2.891592653589793 0.
↪65 1.3 0.0 0.8 0.9 0.0 0.95 0.9 0.0 0.95 40.0 3.116592653589793
pair_coeff * * oxdna2/dh      300.0 0.5 0.815

pair_style hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/
↪dh
pair_coeff * * oxdna2/excv    oxdna2_real.cgdna
pair_coeff * * oxdna2/stk     seqdep 300.0 8.06199211612242 0.005309213 oxdna2_real.cgdna
pair_coeff * * oxdna2/hbond   seqdep oxdna2_real.cgdna
pair_coeff 1 4 oxdna2/hbond   seqdep oxdna2_real.cgdna
pair_coeff 2 3 oxdna2/hbond   seqdep oxdna2_real.cgdna
pair_coeff * * oxdna2/xstk    oxdna2_real.cgdna
pair_coeff * * oxdna2/coaxstk oxdna2_real.cgdna
pair_coeff * * oxdna2/dh      300.0 0.5 oxdna2_real.cgdna
```

Note

The coefficients in the above examples are provided in forms compatible with both *units lj* and *units real* (see documentation of *units*). These can also be read from a potential file with correct unit style by specifying the name of the file. Several potential files for each unit style are included in the potentials directory of the LAMMPS distribution.

4.209.3 Description

The *oxdna2* pair styles compute the pairwise-additive parts of the oxDNA force field for coarse-grained modelling of DNA. The effective interaction between the nucleotides consists of potentials for the excluded volume interaction *oxdna2/excv*, the stacking *oxdna2/stk*, cross-stacking *oxdna2/xstk* and coaxial stacking interaction *oxdna2/coaxstk*, electrostatic Debye-Hueckel interaction *oxdna2/dh* as well as the hydrogen-bonding interaction *oxdna2/hbond* between complementary pairs of nucleotides on opposite strands. Average sequence or sequence-dependent stacking and base-pairing strengths are supported *(Sulc)*. Quasi-unique base-pairing between nucleotides can be achieved by using more complementary pairs of atom types like 5-8 and 6-7, 9-12 and 10-11, 13-16 and 14-15, etc. This prevents the hybridization of in principle complementary bases within Ntypes/4 bases up and down along the backbone.

The exact functional form of the pair styles is rather complex. The individual potentials consist of products of modulation factors, which themselves are constructed from a number of more basic potentials (Morse, Lennard-Jones, harmonic angle and distance) as well as quadratic smoothing and modulation terms. We refer to *(Snodin)* and the original oxDNA publications *(Ouldridge-DPhil)* and *(Ouldridge)* for a detailed description of the oxDNA2 force field.

Note

These pair styles have to be used together with the related oxDNA2 bond style *oxdna2/fene* for the connectivity of the phosphate backbone (see also documentation of *bond_style oxdna2/fene*). Most of the coefficients in the above example have to be kept fixed and cannot be changed without reparameterizing the entire model. Exceptions are the first four coefficients after *oxdna2/stk* (seq=seqdep, T=0.1, xi=1.3523 and kappa=2.6717 and corresponding *real unit* equivalents in the above examples). the first coefficient after *oxdna2/hbond* (seq=seqdep in the above example) and the three coefficients after *oxdna2/dh* (T=0.1, rhos=0.5, qeff=0.815 in the above example). When using a Langevin thermostat e.g. through *fix langevin* or *fix nve/dotc/langevin* the temperature coefficients have to be matched to the one used in the fix.

Note

These pair styles have to be used with the *atom_style hybrid bond ellipsoid oxdna* (see documentation of *atom_style*). The *atom_style oxdna* stores the 3'-to-5' polarity of the nucleotide strand, which is set through the bond topology in the data file. The first (second) atom in a bond definition is understood to point towards the 3'-end (5'-end) of the strand.

Example input and data files for DNA duplexes can be found in examples/PACKAGES/cgdna/examples/oxDNA/ and .../oxDNA2/. A simple python setup tool which creates single straight or helical DNA strands, DNA duplexes or arrays of DNA duplexes can be found in examples/PACKAGES/cgdna/util/.

Please cite *(Henrich)* in any publication that uses this implementation. An updated documentation that contains general information on the model, its implementation and performance as well as the structure of the data and input file can be found here.

Please cite also the relevant oxDNA2 publications *(Snodin)* and *(Sulc)*.

4.209.4 Potential file reading

For each pair style above the first non-modifiable argument can be a filename (with exception of Debye-Hueckel, for which the effective charge argument can be a filename), and if it is, no further arguments should be supplied. Therefore the following command:

```
pair_coeff 1 4 oxdna2/hbond   seqdep oxdna_real.cgdna
```

will be interpreted as a request to read the corresponding hydrogen bonding potential parameters from the file with the given name. The file can define multiple potential parameters for both bonded and pair interactions, but for the example pair interaction above there must exist in the file a line of the form:

```
1 4 hbond     <coefficients>
```

If potential customization is required, the potential file reading can be mixed with the manual specification of the potential parameters. For example, the following command:

```
pair_style hybrid/overlay oxdna2/excv oxdna2/stk oxdna2/hbond oxdna2/xstk oxdna2/coaxstk oxdna2/
↪dh
pair_coeff * * oxdna2/excv    2.0 0.7 0.675 2.0 0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxdna2/stk     seqdep 0.1 1.3523 2.6717 oxdna2_lj.cgdna
pair_coeff * * oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff 1 4 oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff 2 3 oxdna2/hbond   seqdep oxdna2_lj.cgdna
pair_coeff * * oxdna2/xstk    oxdna2_lj.cgdna
pair_coeff * * oxdna2/coaxstk oxdna2_lj.cgdna
pair_coeff * * oxdna2/dh      0.1 0.5 0.815
```

will read the excluded volume and Debye-Hueckel effective charge *qeff* parameters from the manual specification and all others from the potential file *oxdna2_lj.cgdna*.

There are sample potential files for each unit style in the potentials directory of the LAMMPS distribution. The potential file unit system must align with the units defined via the *units* command. For conversion between different *LJ* and *real* unit systems for oxDNA, the python tool *lj2real.py* located in the examples/PACKAGES/cgdna/util/ directory can be used. This tool assumes similar file structure to the examples found in examples/PACKAGES/cgdna/examples/.

4.209.5 Restrictions

These pair styles can only be used if LAMMPS was built with the CG-DNA package and the MOLECULE and ASPHERE package. See the *Build package* page for more info.

4.209.6 Related commands

bond_style oxdna2/fene, *pair_coeff* , *bond_style oxdna/fene*, *pair_style oxdna/excv*, *bond_style oxrna2/fene*, *pair_style oxrna2/excv*, *atom_style oxdna*, *fix nve/dotc/langevin*

4.209.7 Default

none

(Henrich) O. Henrich, Y. A. Gutierrez-Fosado, T. Curk, T. E. Ouldridge, Eur. Phys. J. E 41, 57 (2018).

(Snodin) B.E. Snodin, F. Randisi, M. Mosayebi, et al., J. Chem. Phys. 142, 234901 (2015).

(Sulc) P. Sulc, F. Romano, T.E. Ouldridge, L. Rovigatti, J.P.K. Doye, A.A. Louis, J. Chem. Phys. 137, 135101 (2012).

(Ouldridge-DPhil) T.E. Ouldridge, Coarse-grained modelling of DNA and DNA self-assembly, DPhil. University of Oxford (2011).

(Ouldridge) T.E. Ouldridge, A.A. Louis, J.P.K. Doye, J. Chem. Phys. 134, 085101 (2011).

4.210 pair_style oxrna2/excv command

4.211 pair_style oxrna2/stk command

4.212 pair_style oxrna2/hbond command

4.213 pair_style oxrna2/xstk command

4.214 pair_style oxrna2/coaxstk command

4.215 pair_style oxrna2/dh command

4.215.1 Syntax

```
pair_style style1

pair_coeff * * style2 args
```

- style1 = *hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh*
- style2 = *oxrna2/excv* or *oxrna2/stk* or *oxrna2/hbond* or *oxrna2/xstk* or *oxrna2/coaxstk* or *oxrna2/dh*
- args = list of arguments for these particular styles

oxrna2/stk args = seq T xi kappa 6.0 0.43 0.93 0.35 0.78 0.9 0 0.95 0.9 0 0.95 1.3 0 0.8 1.3 0 0.8 2.0 0.65 2.0 0.65
  seq = seqav (for average sequence stacking strength) or seqdep (for sequence-dependent stacking strength)
  T = temperature (LJ units: 0.1 = 300 K, real units: 300 = 300 K)
  xi = 1.40206 (LJ units) or 8.35864576375849 (real units), temperature-independent coefficient in stacking strength
  kappa = 2.77 (LJ units) or 0.005504556 (real units), coefficient of linear temperature dependence in stacking strength
oxrna2/hbond args = seq eps 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
  seq = seqav (for average sequence base-pairing strength) or seqdep (for sequence-dependent base-pairing strength)
  eps = 0.870439 (LJ units) or 5.18928666388042 (real units), average hydrogen bonding strength between A-U and C-G Watson-Crick and G-U wobble base pairs, 0 between all other pairs
oxrna2/dh args = T rhos qeff
  T = temperature (LJ units: 0.1 = 300 K, real units: 300 = 300 K)
  rhos = salt concentration (mole per litre)
  qeff = 1.02455 (effective charge in elementary charges)

4.215.2 Examples

```
# LJ units
pair_style hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh
pair_coeff * * oxrna2/excv    2.0 0.7 0.675 2.0 0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxrna2/stk     seqdep 0.1 1.40206 2.77 6.0 0.43 0.93 0.35 0.78 0.9 0 0.95 0.9 0 0.95 1.3 0 0.8 1.3 0 0.8 2.0 0.65 2.0 0.65
pair_coeff * * oxrna2/hbond   seqdep 0.0 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 1 4 oxrna2/hbond   seqdep 0.870439 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 2 3 oxrna2/hbond   seqdep 0.870439 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 3 4 oxrna2/hbond   seqdep 0.870439 8.0 0.4 0.75 0.34 0.7 1.5 0 0.7 1.5 0 0.7 1.5 0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff * * oxrna2/xstk    59.9626 0.5 0.6 0.42 0.58 2.25 0.505 0.58 1.7 1.266 0.68 1.7 1.266 0.68 1.7 0.309 0.68 1.7 0.309 0.68
pair_coeff * * oxrna2/coaxstk 80 0.5 0.6 0.42 0.58 2.0 2.592 0.65 1.3 0.151 0.8 0.9 0.685 0.95 0.9 0.685 0.95 2.0 -0.65 2.0 -0.65
pair_coeff * * oxrna2/dh      0.1 0.5 1.02455

pair_style hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh
pair_coeff * * oxrna2/excv    oxrna2_lj.cgdna
pair_coeff * * oxrna2/stk     seqdep 0.1 1.40206 2.77 oxrna2_lj.cgdna
pair_coeff * * oxrna2/hbond   seqdep oxrna2_lj.cgdna
pair_coeff 1 4 oxrna2/hbond   seqdep oxrna2_lj.cgdna
pair_coeff 2 3 oxrna2/hbond   seqdep oxrna2_lj.cgdna
pair_coeff 3 4 oxrna2/hbond   seqdep oxrna2_lj.cgdna
```

(continues on next page)

(continued from previous page)

```
pair_coeff * * oxrna2/xstk    oxrna2_lj.cgdna
pair_coeff * * oxrna2/coaxstk oxrna2_lj.cgdna
pair_coeff * * oxrna2/dh      0.1 0.5 oxrna2_lj.cgdna

# Real units
pair_style hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh
pair_coeff * * oxrna2/excv    11.92337812042065 5.9626 5.74965 11.92337812042065 4.38677 4.259 11.
↪92337812042065 2.81094 2.72576
pair_coeff * * oxrna2/stk     seqdep 300.0 8.35864576375849 0.005504556 0.70439070204273 3.66274 7.
↪92174 2.9813 6.64404 0.9 0.0 0.95 0.9 0.0 0.95 1.3 0.0 0.8 1.3 0.0 0.8 2.0 0.65 2.0 0.65
pair_coeff * * oxrna2/hbond   seqdep 0.0 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.5 0.0 0.7 1.5 0.
↪0 0.7 1.5 0.0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.5707963267948966 0.45
pair_coeff 1 4 oxrna2/hbond   seqdep 5.18928666388042 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.
↪5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.
↪5707963267948966 0.45
pair_coeff 2 3 oxrna2/hbond   seqdep 5.18928666388042 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.
↪5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.
↪5707963267948966 0.45
pair_coeff 3 4 oxrna2/hbond   seqdep 5.18928666388042 0.93918760272364 3.4072 6.3885 2.89612 5.9626 1.
↪5 0.0 0.7 1.5 0.0 0.7 1.5 0.0 0.7 0.46 3.141592653589793 0.7 4.0 1.5707963267948966 0.45 4.0 1.
↪5707963267948966 0.45
pair_coeff * * oxrna2/xstk    4.92690859644113 4.259 5.1108 3.57756 4.94044 2.25 0.505 0.58 1.7 1.266 0.
↪68 1.7 1.266 0.68 1.7 0.309 0.68 1.7 0.309 0.68
pair_coeff * * oxrna2/coaxstk 6.57330882442206 4.259 5.1108 3.57756 4.94044 2.0 2.592 0.65 1.3 0.151 0.
↪8 0.9 0.685 0.95 0.9 0.685 0.95 2.0 -0.65 2.0 -0.65
pair_coeff * * oxrna2/dh      300.0 0.5 1.02455

pair_style hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh
pair_coeff * * oxrna2/excv    oxrna2_real.cgdna
pair_coeff * * oxrna2/stk     seqdep 300.0 8.35864576375849 0.005504556 oxrna2_real.cgdna
pair_coeff * * oxrna2/hbond   seqdep oxrna2_real.cgdna
pair_coeff 1 4 oxrna2/hbond   seqdep oxrna2_real.cgdna
pair_coeff 2 3 oxrna2/hbond   seqdep oxrna2_real.cgdna
pair_coeff 3 4 oxrna2/hbond   seqdep oxrna2_real.cgdna
pair_coeff * * oxrna2/xstk    oxrna2_real.cgdna
pair_coeff * * oxrna2/coaxstk oxrna2_real.cgdna
pair_coeff * * oxrna2/dh      300.0 0.5 oxrna2_real.cgdna
```

Note

The coefficients in the above examples are provided in forms compatible with both *units lj* and *units real* (see documentation of *units*). These can also be read from a potential file with correct unit style by specifying the name of the file. Several potential files for each unit style are included in the potentials directory of the LAMMPS distribution.

4.215.3 Description

The *oxrna2* pair styles compute the pairwise-additive parts of the oxDNA force field for coarse-grained modelling of RNA. The effective interaction between the nucleotides consists of potentials for the excluded volume interaction *oxrna2/excv*, the stacking *oxrna2/stk*, cross-stacking *oxrna2/xstk* and coaxial stacking interaction *oxrna2/coaxstk*, electrostatic Debye-Hueckel interaction *oxrna2/dh* as well as the hydrogen-bonding interaction *oxrna2/hbond* between complementary pairs of nucleotides on opposite strands. Average sequence or sequence-dependent stacking and base-pairing strengths are supported *(Sulc2)*. Quasi-unique base-pairing between nucleotides can be achieved by using more complementary pairs of atom types like 5-8 and 6-7, 9-12 and 10-11, 13-16 and 14-15, etc. This prevents the hybridization of in principle complementary bases within Ntypes/4 bases up and down along the backbone.

The exact functional form of the pair styles is rather complex. The individual potentials consist of products of modulation factors, which themselves are constructed from a number of more basic potentials (Morse, Lennard-Jones, harmonic angle and distance) as well as quadratic smoothing and modulation terms. We refer to *(Sulc1)* and the original oxDNA publications *(Ouldridge-DPhil)* and *(Ouldridge)* for a detailed description of the oxRNA2 force field.

Note

These pair styles have to be used together with the related oxDNA2 bond style *oxrna2/fene* for the connectivity of the phosphate backbone (see also documentation of *bond_style oxrna2/fene*). Most of the coefficients in the above example have to be kept fixed and cannot be changed without reparameterizing the entire model. Exceptions are the first four coefficients after *oxrna2/stk* (seq=seqdep, T=0.1, xi=1.40206 and kappa=2.77 and corresponding *real unit* equivalents in the above examples), the first coefficient after *oxrna2/hbond* (seq=seqdep in the above example) and the three coefficients after *oxrna2/dh* (T=0.1, rhos=0.5, qeff=1.02455 in the above example). When using a Langevin thermostat e.g. through *fix langevin* or *fix nve/dotc/langevin* the temperature coefficients have to be matched to the one used in the fix.

Note

These pair styles have to be used with the *atom_style hybrid bond ellipsoid oxdna* (see documentation of *atom_style*). The *atom_style oxdna* stores the 3'-to-5' polarity of the nucleotide strand, which is set through the bond topology in the data file. The first (second) atom in a bond definition is understood to point towards the 3'-end (5'-end) of the strand.

Example input and data files for DNA duplexes can be found in examples/PACKAGES/cgdna/examples/oxDNA/ and .../oxDNA2/. A simple python setup tool which creates single straight or helical DNA strands, DNA duplexes or arrays of DNA duplexes can be found in examples/PACKAGES/cgdna/util/.

Please cite *(Henrich)* in any publication that uses this implementation. The article contains general information on the model, its implementation and performance as well as the structure of the data and input file. The preprint version of the article can be found here. Please cite also the relevant oxRNA2 publications *(Sulc1)* and *(Sulc2)*.

4.215.4 Potential file reading

For each pair style above the first non-modifiable argument can be a filename (with exception of Debye-Hueckel, for which the effective charge argument can be a filename), and if it is, no further arguments should be supplied. Therefore the following command:

```
pair_coeff 3 4 oxrna2/hbond   seqdep oxrna2_lj.cgdna
```

will be interpreted as a request to read the corresponding hydrogen bonding potential parameters from the file with the given name. The file can define multiple potential parameters for both bonded and pair interactions, but for the example pair interaction above there must exist in the file a line of the form:

```
3 4 hbond     <coefficients>
```

If potential customization is required, the potential file reading can be mixed with the manual specification of the potential parameters. For example, the following command:

```
pair_style hybrid/overlay oxrna2/excv oxrna2/stk oxrna2/hbond oxrna2/xstk oxrna2/coaxstk oxrna2/dh
pair_coeff * * oxrna2/excv     2.0 0.7 0.675 2.0 0.515 0.5 2.0 0.33 0.32
pair_coeff * * oxrna2/stk      seqdep 0.1 1.40206 2.77 oxrna2_lj.cgdna
pair_coeff * * oxrna2/hbond    seqdep oxrna2_lj.cgdna
pair_coeff 1 4 oxrna2/hbond    seqdep oxrna2_lj.cgdna
pair_coeff 2 3 oxrna2/hbond    seqdep oxrna2_lj.cgdna
pair_coeff 3 4 oxrna2/hbond    seqdep oxrna2_lj.cgdna
pair_coeff * * oxrna2/xstk     oxrna2_lj.cgdna
pair_coeff * * oxrna2/coaxstk oxrna2_lj.cgdna
pair_coeff * * oxrna2/dh       0.1 0.5 1.02455
```

will read the excluded volume and Debye-Hueckel effective charge *qeff* parameters from the manual specification and all others from the potential file *oxrna2_lj.cgdna*.

There are sample potential files for each unit style in the potentials directory of the LAMMPS distribution. The potential file unit system must align with the units defined via the *units* command. For conversion between different *LJ* and *real* unit systems for oxDNA, the python tool *lj2real.py* located in the examples/PACKAGES/cgdna/util/ directory can be used. This tool assumes similar file structure to the examples found in examples/PACKAGES/cgdna/examples/.

4.215.5 Restrictions

These pair styles can only be used if LAMMPS was built with the CG-DNA package and the MOLECULE and ASPHERE package. See the *Build package* page for more info.

4.215.6 Related commands

bond_style oxrna2/fene, *pair_coeff*, *bond_style oxdna/fene*, *pair_style oxdna/excv*, *bond_style oxdna2/fene*, *pair_style oxdna2/excv*, *atom_style oxdna*, *fix nve/dotc/langevin*

4.215.7 Default

none

(Henrich) O. Henrich, Y. A. Gutierrez-Fosado, T. Curk, T. E. Ouldridge, Eur. Phys. J. E 41, 57 (2018).

(Sulc1) P. Sulc, F. Romano, T. E. Ouldridge, et al., J. Chem. Phys. 140, 235102 (2014).

(Sulc2) P. Sulc, F. Romano, T.E. Ouldridge, L. Rovigatti, J.P.K. Doye, A.A. Louis, J. Chem. Phys. 137, 135101 (2012).

(Ouldridge-DPhil) T.E. Ouldridge, Coarse-grained modelling of DNA and DNA self-assembly, DPhil. University of Oxford (2011).

(Ouldridge) T.E. Ouldridge, A.A. Louis, J.P.K. Doye, J. Chem. Phys. 134, 085101 (2011).

4.216 pair_style pace command

Accelerator Variants: *pace/kk*, *pace/extrapolation/kk*

4.217 pair_style pace/extrapolation command

4.217.1 Syntax

```
pair_style pace ... keyword values ...
```

- one or more keyword/value pairs may be appended

  keyword = product or recursive or chunksize
    product = use product algorithm for basis functions
    recursive = use recursive algorithm for basis functions
    chunksize value = number of atoms in each pass

```
pair_style pace/extrapolation
```

4.217.2 Examples

```
pair_style pace
pair_style pace product chunksize 2048
pair_coeff * * Cu-PBE-core-rep.ace Cu

pair_style pace
pair_coeff * * Cu.yaml Cu

pair_style pace/extrapolation
pair_coeff * * Cu.yaml Cu.asi Cu
```

4.217.3 Description

Pair style *pace* computes interactions using the Atomic Cluster Expansion (ACE), which is a general expansion of the atomic energy in multi-body basis functions. *(Drautz19)*. The *pace* pair style provides an efficient implementation that is described in this paper *(Lysogorskiy21)*.

In ACE, the total energy is decomposed into a sum over atomic energies. The energy of atom *i* is expressed as a linear or non-linear function of one or more density functions. By projecting the density onto a local atomic base, the lowest order contributions to the energy can be expressed as a set of scalar polynomials in basis function contributions summed over neighbor atoms.

Only a single pair_coeff command is used with the *pace* style which specifies an ACE coefficient file followed by N additional arguments specifying the mapping of ACE elements to LAMMPS atom types, where N is the number of LAMMPS atom types:

- ACE coefficient file (.yaml or .yace/.ace format)
- N element names = mapping of ACE elements to atom types

Only a single pair_coeff command is used with the *pace* style which specifies an ACE file that fully defines the potential. Note that unlike for other potentials, cutoffs are not set in the pair_style or pair_coeff command; they are specified in the ACE file.

The pair_style *pace* command may be followed by the optional keyword *product* or *recursive*, which determines which of two algorithms is used for the calculation of basis functions and derivatives. The default is *recursive*.

The keyword *chunksize* is only applicable when using the pair style *pace* with the KOKKOS package on GPUs and is ignored otherwise. This keyword controls the number of atoms in each pass used to compute the atomic cluster expansion and is used to avoid running out of memory. For example if there are 8192 atoms in the simulation and the *chunksize* is set to 4096, the ACE calculation will be broken up into two passes (running on a single GPU).

4.217.4 Extrapolation grade

Calculation of extrapolation grade in PACE is implemented in *pair_style pace/extrapolation*. It is based on the MaxVol algorithm similar to Moment Tensor Potential (MTP) by Shapeev et al. and is described in *(Lysogorskiy23)*. In order to compute extrapolation grade one needs to provide:

1. ACE potential in B-basis form (*.yaml* format) and
2. Active Set Inverted (ASI) file for corresponding potential (*.asi* format)

Calculation of extrapolation grades requires matrix-vector multiplication for each atom and is slower than the usual *pair_style pace recursive*, therefore it is *not* computed by default. Extrapolation grade calculation is involved by *fix pair*, which requests to compute *gamma*, as shown in example below:

```
pair_style  pace/extrapolation
pair_coeff  * * Cu.yaml Cu.asi Cu

fix pace_gamma all pair 10 pace/extrapolation gamma 1

compute max_pace_gamma all reduce max f_pace_gamma
variable dump_skip equal "c_max_pace_gamma < 5"

dump pace_dump all custom 20 extrapolative_structures.dump id type x y z f_pace_gamma
dump_modify pace_dump skip v_dump_skip

variable max_pace_gamma equal c_max_pace_gamma
fix extreme_extrapolation all halt 10 v_max_pace_gamma > 25
```

Here extrapolation grade gamma is computed every 10 steps and is stored in *f_pace_gamma* per-atom variable. The largest value of extrapolation grade among all atoms in a structure is reduced to *c_max_pace_gamma* variable. Only if this value exceeds extrapolation threshold 5, then the structure will be dumped into *extrapolative_structures.dump* file, but not more often than every 20 steps.

On all other steps *pair_style pace recursive* will be used.

When using the pair style *pace/extrapolation* with the KOKKOS package on GPUs product B-basis evaluator is always used and only *linear* ASI is supported.

See the *pair_coeff* page for alternate ways to specify the path for the ACE coefficient file.

4.217.5 Core repulsion

The ACE potential can be configured to initiate core-repulsion from an inner cutoff, seamlessly transitioning from ACE to ZBL. The core repulsion factor can be accessed as a per-atom quantity, as demonstrated in the example below:

```
pair_style  pace
pair_coeff  * * CuNi.yaml Cu Ni

fix pace_corerep all pair 1 pace corerep 1
```

In this case, per-atom *f_pace_corerep* quantities represent the fraction of ZBL core-repulsion for each atom.

4.217.6 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS with user-specifiable parameters as described above. You never need to specify a pair_coeff command with I != J arguments for this style.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.217.7 Restrictions

This pair style is part of the ML-PACE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.217.8 Related commands

pair_style snap, *fix pair*

4.217.9 Default

recursive, chunksize = 4096,

(Drautz19) Drautz, Phys Rev B, 99, 014104 (2019).

(Lysogorskiy21) Lysogorskiy, van der Oord, Bochkarev, Menon, Rinaldi, Hammerschmidt, Mrovec, Thompson, Csanyi, Ortner, Drautz, npj Comp Mat, 7, 97 (2021).

(Lysogorskiy23) Lysogorskiy, Bochkarev, Mrovec, Drautz, Phys Rev Mater, 7, 043801 (2023) / arXiv:2212.08716 (2022).

4.218 pair_style pedone command

Accelerator Variants: *pedone/omp*

4.218.1 Syntax

```
pair_style style args
```

- style = pedone*
- args = list of arguments for a particular style

```
pedone args = cutoff
  cutoff = global cutoff for Pedone interactions (distance units)
```

4.218.2 Examples

```
pair_style hybrid/overlay pedone 15.0 coul/long 15.0
kspace_style pppm 1.0e-5

pair_coeff * * coul/long
pair_coeff 1 2 pedone 0.030211 2.241334 2.923245 5.0
pair_coeff 2 2 pedone 0.042395 1.379316 3.618701 22.0
```

Used in input scripts:

```
examples/PACKAGES/pedone/in.pedone.relax
examples/PACKAGES/pedone/in.pedone.melt
```

4.218.3 Description

Added in version 17Apr2024.

Pair style *pedone* computes the **non-Coulomb** interactions of the Pedone (or PMMCS) potential *(Pedone)* which combines Coulomb interactions, Morse potential, and repulsive r^{-12} Lennard-Jones terms (see below). The *pedone* pair style is meant to be used in addition to a *Coulomb pair style* via pair style *hybrid/overlay* (see example above). Using *coul/long* or *could/dsf* (for solids) is recommended.

The full Pedone potential function from *(Pedone)* for each pair of atoms is:

$$E = \frac{Cq_iq_j}{\varepsilon r} + D_0\left[e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}\right] + \frac{B_0}{r^{12}} \qquad r < r_c$$

r_c is the cutoff and C is a conversion factor that is specific to the choice of *units* so that the entire Coulomb term is in energy units with q_i and q_j as the assigned charges in multiples of the elementary charge.

The following coefficients must be defined for the selected pairs of atom types via the *pair_coeff* command as in the example above:

- D_0 (energy units)
- α (1/distance units)
- r_0 (distance units)
- C_0 (energy units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global *pedone* cutoff is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.218.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing.

This pair style support the *pair_modify* shift option for the energy of the pair interaction.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands does not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, or *outer* keywords.

4.218.5 Restrictions

The *pedone* pair style is only enabled if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* page for more info.

4.218.6 Related commands

pair_coeff , *pair_style*, *pair style coul/long and coul/dsf* , *pair style morse*

4.218.7 Default

none

(Pedone) A. Pedone, G. Malavasi, M. C. Menziani, A. N. Cormack, and U. Segre, J. Phys. Chem. B, 110, 11780 (2006)

4.219 pair_style peri/pmb command

Accelerator Variants: *peri/pmb/omp*

4.220 pair_style peri/lps command

Accelerator Variants: *peri/lps/omp*

4.221 pair_style peri/ves command

4.222 pair_style peri/eps command

4.222.1 Syntax

```
pair_style style
```

- style = *peri/pmb* or *peri/lps* or *peri/ves* or *peri/eps*

4.222.2 Examples

```
pair_style peri/pmb
pair_coeff * * 1.6863e22 0.0015001 0.0005 0.25

pair_style peri/lps
pair_coeff * * 14.9e9 14.9e9 0.0015001 0.0005 0.25

pair_style peri/ves
```

(continues on next page)

(continued from previous page)

```
pair_coeff * * 14.9e9 14.9e9 0.0015001 0.0005 0.25 0.5 0.001

pair_style peri/eps
pair_coeff * * 14.9e9 14.9e9 0.0015001 0.0005 0.25 118.43
```

4.222.3 Description

The peridynamic pair styles implement material models that can be used at the mesoscopic and macroscopic scales. See this document for an overview of LAMMPS commands for Peridynamics modeling.

Style *peri/pmb* implements the Peridynamic bond-based prototype microelastic brittle (PMB) model.

Style *peri/lps* implements the Peridynamic state-based linear peridynamic solid (LPS) model.

Style *peri/ves* implements the Peridynamic state-based linear peridynamic viscoelastic solid (VES) model.

Style *peri/eps* implements the Peridynamic state-based elastic-plastic solid (EPS) model.

The canonical papers on Peridynamics are *(Silling 2000)* and *(Silling 2007)*. The implementation of Peridynamics in LAMMPS is described in *(Parks)*. Also see the *Peridynamics Howto* for more details about its implementation.

The peridynamic VES and EPS models in PDLAMMPS were implemented by R. Rahman and J. T. Foster at University of Texas at San Antonio. The original VES formulation is described in "(Mitchell2011)" and the original EPS formulation is in "(Mitchell2011a)". Additional PDF docs that describe the VES and EPS implementations are include in the LAMMPS distribution in doc/PDF/PDLammps_VES.pdf and doc/PDF/PDLammps_EPS.pdf. For questions regarding the VES and EPS models in LAMMPS you can contact R. Rahman (rezwanur.rahman at utsa.edu).

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below.

For the *peri/pmb* style:

- c (energy/distance/volume^2 units)
- horizon (distance units)
- s00 (unitless)
- α (unitless)

C is the effectively a spring constant for Peridynamic bonds, the horizon is a cutoff distance for truncating interactions, and s00 and α are used as a bond breaking criteria. The units of c are such that c/distance = stiffness/volume^2, where stiffness is energy/distance^2 and volume is distance^3. See the users guide for more details.

For the *peri/lps* style:

- K (force/area units)
- G (force/area units)
- horizon (distance units)
- s00 (unitless)
- α (unitless)

K is the bulk modulus and G is the shear modulus. The horizon is a cutoff distance for truncating interactions, and s00 and α are used as a bond breaking criteria. See the users guide for more details.

For the *peri/ves* style:

- K (force/area units)
- G (force/area units)
- horizon (distance units)
- s00 (unitless)
- α (unitless)
- m_lambdai (unitless)
- m_taubi (unitless)

K is the bulk modulus and G is the shear modulus. The horizon is a cutoff distance for truncating interactions, and s00 and α are used as a bond breaking criteria. m_lambdai and m_taubi are the viscoelastic relaxation parameter and time constant, respectively. m_lambdai varies within zero to one. For very small values of m_lambdai the viscoelastic model responds very similar to a linear elastic model. For details please see the description in "(Mitchell2011)".

For the *peri/eps* style:

- K (force/area units)
- G (force/area units)
- horizon (distance units)
- s00 (unitless)
- α (unitless)
- m_yield_stress (force/area units)

K is the bulk modulus and G is the shear modulus. The horizon is a cutoff distance and s00 and α are used as a bond breaking criteria. m_yield_stress is the yield stress of the material. For details please see the description in "(Mitchell2011a)".

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.222.4 Mixing, shift, table, tail correction, restart, rRESPA info

These pair styles do not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

These pair styles do not support the *pair_modify* shift option.

The *pair_modify* table and tail options are not relevant for these pair styles.

These pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

These pair styles can only be used via the *pair* keyword of the *run_style respa* command. They do not support the *inner*, *middle*, *outer* keywords.

4.222.5 Restrictions

All of these styles are part of the PERI package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.222.6 Related commands

pair_coeff

4.222.7 Default

none

(Parks) Parks, Lehoucq, Plimpton, Silling, Comp Phys Comm, 179(11), 777-783 (2008).

(Silling 2000) Silling, J Mech Phys Solids, 48, 175-209 (2000).

(Silling 2007) Silling, Epton, Weckner, Xu, Askari, J Elasticity, 88, 151-184 (2007).

(Mitchell2011) Mitchell. A non-local, ordinary-state-based viscoelasticity model for peridynamics. Sandia National Lab Report, 8064:1-28 (2011).

(Mitchell2011a) Mitchell. A Nonlocal, Ordinary, State-Based Plasticity Model for Peridynamics. Sandia National Lab Report, 3166:1-34 (2011).

4.223 pair_style pod command

Accelerator Variants: *pod/kk*

4.223.1 Syntax

```
pair_style pod
```

4.223.2 Examples

```
pair_style pod
pair_coeff * * Ta_param.pod Ta_coefficients.pod Ta
```

4.223.3 Description

Added in version 22Dec2022.

Pair style *pod* defines the proper orthogonal descriptor (POD) potential *(Nguyen and Rohskopf)*, *(Nguyen2023)*, *(Nguyen2024)*, and *(Nguyen and Sema)*. The *fitpod* is used to fit the POD potential.

Only a single pair_coeff command is used with the *pod* style which specifies a POD parameter file followed by a coefficient file, a projection matrix file, and a centroid file.

The POD parameter file (Ta_param.pod) can contain blank and comment lines (start with #) anywhere. Each non-blank non-comment line must contain one keyword/value pair. See *fitpod* for the description of all the keywords that can be assigned in the parameter file.

The coefficient file (Ta_coefficients.pod) contains coefficients for the POD potential. The top of the coefficient file can contain any number of blank and comment lines (start with #), but follows a strict format after that. The first non-blank non-comment line must contain:

- model_coefficients: *ncoeff nproj ncentroid*

This is followed by *ncoeff* coefficients, *nproj* projection matrix entries, and *ncentroid* centroid coordinates, one per line. The coefficient file is generated after training the POD potential using *fitpod*.

As an example, if a LAMMPS indium phosphide simulation has 4 atoms types, with the first two being indium and the third and fourth being phophorous, the pair_coeff command would look like this:

```
pair_coeff * * pod InP_param.pod InP_coefficients.pod In In P P
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The two filenames are for the parameter and coefficient files, respectively. The two trailing 'In' arguments map LAMMPS atom types 1 and 2 to the POD 'In' element. The two trailing 'P' arguments map LAMMPS atom types 3 and 4 to the POD 'P' element.

If a POD mapping value is specified as NULL, the mapping is not performed. This can be used when a *pod* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Examples about training and using POD potentials are found in the directory lammps/examples/PACKAGES/pod and the Github repo https://github.com/cesmix-mit/pod-examples.

4.223.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS with user-specifiable parameters as described above. You never need to specify a pair_coeff command with I != J arguments for this style.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.223.5 Restrictions

This style is part of the ML-POD package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.223.6 Related commands

fitpod, *compute pod/atom*, *compute podd/atom*, *compute pod/local*, *compute pod/global*

4.223.7 Default

none

(Nguyen and Rohskopf) Nguyen and Rohskopf, Journal of Computational Physics, 480, 112030, (2023).

(Nguyen2023) Nguyen, Physical Review B, 107(14), 144103, (2023).

(Nguyen2024) Nguyen, Journal of Computational Physics, 113102, (2024).

(Nguyen and Sema) Nguyen and Sema, https://arxiv.org/abs/2405.00306, (2024).

4.224 pair_style polymorphic command

4.224.1 Syntax

```
pair_style polymorphic
```

style = *polymorphic*

4.224.2 Examples

```
pair_style polymorphic
pair_coeff * * FeCH_BOP_I.poly Fe C H
pair_coeff * * TlBr_msw.poly Tl Br
pair_coeff * * CuTa_eam.poly Cu Ta
pair_coeff * * GaN_tersoff.poly Ga N
pair_coeff * * GaN_sw.poly Ga N
```

4.224.3 Description

The *polymorphic* pair style computes a 3-body free-form potential (*Zhou3*) for the energy E of a system of atoms as

$$E = \frac{1}{2}\sum_{i=1}^{i=N}\sum_{j=1}^{j=N}\left[\left(1-\delta_{ij}\right)\cdot U_{IJ}\left(r_{ij}\right) - \left(1-\eta_{ij}\right)\cdot F_{IJ}\left(X_{ij}\right)\cdot V_{IJ}\left(r_{ij}\right)\right]$$

$$X_{ij} = \sum_{k=i_1,k\neq j}^{i_N} W_{IK}\left(r_{ik}\right)\cdot G_{JIK}\left(\cos\theta_{jik}\right)\cdot P_{JIK}\left(\Delta r_{jik}\right)$$

$$\Delta r_{jik} = r_{ij} - \xi_{IJ}\cdot r_{ik}$$

where I, J, K represent species of atoms i, j, and k, $i_1, ..., i_N$ represents a list of i's neighbors, δ_{ij} is a Dirac constant (i.e., $\delta_{ij} = 1$ when $i = j$, and $\delta_{ij} = 0$ otherwise), η_{ij} is similar constant that can be set either to $\eta_{ij} = \delta_{ij}$ or $\eta_{ij} = 1 - \delta_{ij}$ depending on the potential type, $U_{IJ}(r_{ij})$, $V_{IJ}(r_{ij})$, $W_{IK}(r_{ik})$ are pair functions, $G_{JIK}(\cos\theta_{jik})$ is an angular function, $P_{JIK}(\Delta r_{jik})$ is a function of atomic spacing differential $\Delta r_{jik} = r_{ij} - \xi_{IJ}\cdot r_{ik}$ with ξ_{IJ} being a pair-dependent parameter, and $F_{IJ}(X_{ij})$ is a function of the local environment variable X_{ij}. This generic potential is fully defined once the constants η_{ij} and ξ_{IJ}, and the six functions $U_{IJ}(r_{ij})$, $V_{IJ}(r_{ij})$, $W_{IK}(r_{ik})$, $G_{JIK}(\cos\theta_{jik})$, $P_{JIK}(\Delta r_{jik})$, and $F_{IJ}(X_{ij})$ are given. Here LAMMPS uses a global parameter η to represent η_{ij}. When $\eta = 1$, $\eta_{ij} = 1 - \delta_{ij}$, otherwise $\eta_{ij} = \delta_{ij}$. Additionally, $\eta = 3$ indicates that the function $P_{JIK}(\Delta r)$ depends on species I, J and K, otherwise $P_{JIK}(\Delta r) = P_{IK}(\Delta r)$ only depends on species I and K. Note that these six functions are all one dimensional, and hence can be provided in a tabular form. This allows users to design different potentials solely based on a manipulation of these functions. For instance, the potential reduces to a Stillinger-Weber potential (*SW*) if we set

$$\eta_{ij} = \delta_{ij}(\eta = 2\ or\ \eta = 0), \xi_{IJ} = 0$$

$$U_{IJ}(r) = A_{IJ}\cdot\varepsilon_{IJ}\cdot\left(\frac{\sigma_{IJ}}{r}\right)^{q}\cdot\left[B_{IJ}\cdot\left(\frac{\sigma_{IJ}}{r}\right)^{p-q} - 1\right]\cdot exp\left(\frac{\sigma_{IJ}}{r - a_{IJ}\cdot\sigma_{IJ}}\right)$$

$$V_{IJ}(r) = \sqrt{\lambda_{IJ}\cdot\varepsilon_{IJ}}\cdot exp\left(\frac{\gamma_{IJ}\cdot\sigma_{IJ}}{r - a_{IJ}\cdot\sigma_{IJ}}\right)$$

$$F_{IJ}(X) = -X$$

$$P_{JIK}(\Delta r) = P_{IK}(\Delta r) = 1$$

$$W_{IJ}(r) = \sqrt{\lambda_{IJ}\cdot\varepsilon_{IJ}}\cdot exp\left(\frac{\gamma_{IJ}\cdot\sigma_{IJ}}{r - a_{IJ}\cdot\sigma_{IJ}}\right)$$

$$G_{JIK}(\cos\theta) = \left(\cos\theta + \frac{1}{3}\right)^{2}$$

The potential reduces to a Tersoff potential (*Tersoff* or *Albe1*) if we set

$$\eta_{ij} = \delta_{ij}(\eta = 2\ or\ \eta = 0), \xi_{IJ} = 1$$

$$U_{IJ}(r) = \frac{D_{e,IJ}}{S_{IJ} - 1}\cdot exp\left[-\beta_{IJ}\sqrt{2S_{IJ}}\left(r - r_{e,IJ}\right)\right]\cdot f_{c,IJ}(r)$$

$$V_{IJ}(r) = \frac{S_{IJ}\cdot D_{e,IJ}}{S_{IJ} - 1}\cdot exp\left[-\beta_{IJ}\sqrt{\frac{2}{S_{IJ}}}\left(r - r_{e,IJ}\right)\right]\cdot f_{c,IJ}(r)$$

$$F_{IJ}(X) = (1+X)^{-\frac{1}{2}}$$

$$P_{JIK}(\Delta r) = P_{IK}(\Delta r) = exp(2\mu_{IK}\cdot\Delta r)$$

$$W_{IJ}(r) = f_{c,IJ}(r)$$

$$G_{JIK}(\cos\theta) = \gamma_{IK}\left[1 + \frac{c_{IK}^2}{d_{IK}^2} - \frac{c_{IK}^2}{d_{IK}^2 + (h_{IK} + \cos\theta)^2}\right]$$

where

$$f_{c,IJ}(r) = \begin{cases} 1, r \leq R_{IJ} - D_{IJ} \\ \frac{1}{2} + \frac{1}{2}cos\left[\frac{\pi(r+D_{IJ}-R_{IJ})}{2D_{IJ}}\right], R_{IJ} - D_{IJ} < r < R_{IJ} + D_{IJ} \\ 0, r \geq R_{IJ} + D_{IJ} \end{cases}$$

The potential reduces to a modified Stillinger-Weber potential (*Zhou3*) if we set

$$\begin{aligned}
\eta_{ij} &= \delta_{ij}(\eta = 2 \; or \; \eta = 0), \xi_{IJ} = 0 \\
U_{IJ}(r) &= \varphi_{R,IJ}(r) - \varphi_{A,IJ}(r) \\
V_{IJ}(r) &= u_{IJ}(r) \\
F_{IJ}(X) &= -X \\
P_{JIK}(\Delta r) &= P_{IK}(\Delta r) = 1 \\
W_{IJ}(r) &= u_{IJ}(r) \\
G_{JIK}(\cos\theta) &= g_{JIK}(\cos\theta)
\end{aligned}$$

The potential reduces to a Rockett-Tersoff potential (*Wang3*) if we set

$$\begin{aligned}
\eta_{ij} &= \delta_{ij}(\eta = 2 \; or \; \eta = 0), \xi_{IJ} = 1 \\
U_{IJ}(r) &= A_{IJ} exp\left(-\lambda_{1,IJ} \cdot r\right) f_{c,IJ}(r) f_{ca,IJ}(r) \\
V_{IJ}(r) &= \left\{ \begin{array}{l} B_{IJ} exp\left(-\lambda_{2,IJ} \cdot r\right) f_{c,IJ}(r) + \\ A_{IJ} exp\left(-\lambda_{1,IJ} \cdot r\right) f_{c,IJ}(r) \left[1 - f_{ca,IJ}(r)\right] \end{array} \right\} \\
F_{IJ}(X) &= \left[1 + \left(\beta_{IJ} X\right)^{n_{IJ}}\right]^{-\frac{1}{2n_{IJ}}} \\
P_{JIK}(\Delta r) &= P_{IK}(\Delta r) = exp\left(\lambda_{3,IK} \cdot \Delta r^3\right) \\
W_{IJ}(r) &= f_{c,IJ}(r) \\
G_{JIK}(\cos\theta) &= 1 + \frac{c_{IK}^2}{d_{IK}^2} - \frac{c_{IK}^2}{d_{IK}^2 + (h_{IK} + \cos\theta)^2}
\end{aligned}$$

where $f_{ca,IJ}(r)$ is similar to the $f_{c,IJ}(r)$ defined above:

$$f_{ca,IJ}(r) = \left\{ \begin{array}{l} 1, r \leq R_{a,IJ} - D_{a,IJ} \\ \frac{1}{2} + \frac{1}{2} cos\left[\frac{\pi\left(r + D_{a,IJ} - R_{a,IJ}\right)}{2D_{a,IJ}}\right], R_{a,IJ} - D_{a,IJ} < r < R_{a,IJ} + D_{a,IJ} \\ 0, r \geq R_{a,IJ} + D_{a,IJ} \end{array} \right.$$

The potential becomes the embedded atom method (*Daw*) if we set

$$\begin{aligned}
\eta_{ij} &= 1 - \delta_{ij}(\eta = 1), \xi_{IJ} = 0 \\
U_{IJ}(r) &= \phi_{IJ}(r) \\
V_{IJ}(r) &= 1 \\
F_{II}(X) &= -2F_I(X) \\
P_{JIK}(\Delta r) &= P_{IK}(\Delta r) = 1 \\
W_{IJ}(r) &= f_J(r) \\
G_{JIK}(\cos\theta) &= 1
\end{aligned}$$

In the embedded atom method case, $\phi_{IJ}(r)$ is the pair energy, $F_I(X)$ is the embedding energy, X is the local electron density, and $f_J(r)$ is the atomic electron density function.

The potential reduces to another type of Tersoff potential (*Zhou4*) if we set

$$\eta_{ij} = \delta_{ij}(\eta = 3), \xi_{IJ} = 1$$

$$U_{IJ}(r) = \frac{D_{e,IJ}}{S_{IJ} - 1} \cdot exp\left[-\beta_{IJ}\sqrt{2S_{IJ}}\left(r - r_{e,IJ}\right)\right] \cdot f_{c,IJ}(r) \cdot T_{IJ}(r) + V_{ZBL,IJ}(r)\left[1 - T_{IJ}(r)\right]$$

$$V_{IJ}(r) = \frac{S_{IJ} \cdot D_{e,IJ}}{S_{IJ} - 1} \cdot exp\left[-\beta_{IJ}\sqrt{\frac{2}{S_{IJ}}}\left(r - r_{e,IJ}\right)\right] \cdot f_{c,IJ}(r) \cdot T_{IJ}(r)$$

$$F_{IJ}(X) = (1 + X)^{-\frac{1}{2}}$$

$$P_{JIK}(\Delta r) = \omega_{JIK} \cdot exp(\alpha_{JIK} \cdot \Delta r)$$

$$W_{IJ}(r) = f_{c,IJ}(r)$$

$$G_{JIK}(\cos\theta) = \gamma_{JIK}\left[1 + \frac{c_{JIK}^2}{d_{JIK}^2} - \frac{c_{JIK}^2}{d_{JIK}^2 + (h_{JIK} + \cos\theta)^2}\right]$$

$$T_{IJ}(r) = \frac{1}{1 + exp\left[-b_{f,IJ}\left(r - r_{f,IJ}\right)\right]}$$

$$V_{ZBL,IJ}(r) = 14.4 \cdot \frac{Z_I \cdot Z_J}{r} \sum_{k=1}^{4} \mu_k \cdot exp\left[-\nu_k\left(Z_I^{0.23} + Z_J^{0.23}\right) r\right]$$

where $f_{c,IJ}(r)$ is the same as defined above. This Tersoff potential differs from the one above because the $P_{JIK}(\Delta r)$ function is now dependent on all three species I, J, and K.

If the tabulated functions are created using the parameters of Stillinger-Weber, Tersoff, and EAM potentials, the polymorphic pair style will produce the same global properties (energies and stresses) and the same forces as the *sw*, *tersoff*, and *eam* pair styles. The polymorphic pair style also produces the same per-atom properties (energies and stresses) as the corresponding *tersoff* and *eam* pair styles. However, due to a different partitioning of global properties to per-atom properties, the polymorphic pair style will produce different per-atom properties (energies and stresses) as the *sw* pair style. This does not mean that polymorphic pair style is different from the sw pair style. It just means that the definitions of the atom energies and atom stresses are different.

Only a single pair_coeff command is used with the polymorphic pair style which specifies a potential file for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of polymorphic potential elements to atom types

See the pair_coeff page for alternate ways to specify the path for the potential file. Several files for polymorphic potentials are included in the potentials directory of the LAMMPS distribution. They have a "poly" suffix.

As an example, imagine the GaN_tersoff.poly file has tabulated functions for Ga-N tersoff potential. If your LAMMPS simulation has 4 atom types and you want the first 3 to be Ga, and the fourth to be N, you would use the following pair_coeff command:

```
pair_coeff * * GaN_tersoff.poly Ga Ga Ga N
```

The first two arguments must be * * to span all pairs of LAMMPS atom types. The first three Ga arguments map LAMMPS atom types 1,2,3 to the Ga element in the polymorphic file. The final N argument maps LAMMPS atom type 4 to the N element in the polymorphic file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when an polymorphic potential is used as part of the hybrid pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Potential files in the potentials directory of the LAMMPS distribution have a ".poly" suffix. At the beginning of the files, an unlimited number of lines starting with '#' are used to describe the potential and are ignored by LAMMPS. The next line lists two numbers:

```
ntypes eta
```

Here *ntypes* represent total number of species defined in the potential file, η = 1 reduces to embedded atom method, η = 3 assumes a three species dependent $P_{JIK}(\Delta r)$ function, and all other values of η assume a two species dependent $P_{JK}(\Delta r)$ function. The value of *ntypes* must equal the total number of different species defined in the pair_coeff command. The next *ntypes* lines each lists two numbers and a character string representing atomic number, atomic mass, and name of the species of the ntypes elements:

```
atomic-number atomic-mass element-name(1)
atomic-number atomic-mass element-name(2)
...
atomic-number atomic-mass element-name(ntypes)
```

The next line contains four numbers:

```
nr ntheta nx xmax
```

Here nr is total number of tabular points for radial functions U, V, W, P, ntheta is total number of tabular points for the angular function G, nx is total number of tabular points for the function F, xmax is a maximum value of the argument of function F. Note that the pair functions $U_{IJ}(r)$, $V_{IJ}(r)$, $W_{IJ}(r)$ are uniformly tabulated between 0 and cutoff distance of the IJ pair, $G_{JIK}(\cos\theta)$ is uniformly tabulated between -1 and 1, $P_{JIK}(\Delta r)$ is uniformly tabulated between -rcmax and rcmax where rcmax is the maximum cutoff distance of all pairs, and $F_{IJ}(X)$ is uniformly tabulated between 0 and xmax. Linear extrapolation is assumed if actual simulations exceed these ranges.

The next ntypes*(ntypes+1)/2 lines contain two numbers:

cut xi(1)
cut xi(2)
...
cut xi(ntypes*(ntypes+1)/2)

Here cut means the cutoff distance of the pair functions, "xi" is ξ as defined in the potential functions above. The ntypes*(ntypes+1)/2 lines are related to the pairs according to the sequence of first ii (self) pairs, i = 1, 2, …, ntypes, and then ij (cross) pairs, i = 1, 2, …, ntypes-1, and j = i+1, i+2, …, ntypes (i.e., the sequence of the ij pairs follows 11, 22, …, 12, 13, 14, …, 23, 24, …).

In the final blocks of the potential file, U, V, W, P, G, and F functions are listed sequentially. First, U functions are given for each of the ntypes*(ntypes+1)/2 pairs according to the sequence described above. For each of the pairs, nr values are listed. Next, similar arrays are given for V and W functions. If P functions depend only on pair species, i.e., $\eta \neq 3$, then P functions are also listed the same way the next. If P functions depend on three species, i.e., η = 3, then P functions are listed for all the ntypes*ntypes*ntypes IJK triplets in a natural sequence I from 1 to ntypes, J from 1 to ntypes, and K from 1 to ntypes (i.e., IJK = 111, 112, 113, …, 121, 122, 123 …, 211, 212, …). Next, G functions are listed for all the ntypes*ntypes*ntypes IJK triplets similarly. For each of the G functions, ntheta values are listed. Finally, F functions are listed for all the ntypes*(ntypes+1)/2 pairs in the same sequence as described above. For each of the F functions, nx values are listed.

4.224.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write their information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.224.5 Restrictions

If using create_atoms command, atomic masses must be defined in the input script. If using read_data, atomic masses must be defined in the atomic structure data file.

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair potential requires the *newton* setting to be "on" for pair interactions.

The potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use any LAMMPS units, but you would need to create your own potential files.

4.224.6 Related commands

pair_coeff

(Zhou3) X. W. Zhou, M. E. Foster, R. E. Jones, P. Yang, H. Fan, and F. P. Doty, J. Mater. Sci. Res., 4, 15 (2015).

(Zhou4) X. W. Zhou, M. E. Foster, J. A. Ronevich, and C. W. San Marchi, J. Comp. Chem., 41, 1299 (2020).

(SW) F. H. Stillinger, and T. A. Weber, Phys. Rev. B, 31, 5262 (1985).

(Tersoff) J. Tersoff, Phys. Rev. B, 39, 5566 (1989).

(Albe1) K. Albe, K. Nordlund, J. Nord, and A. Kuronen, Phys. Rev. B, 66, 035205 (2002).

(Wang) J. Wang, and A. Rockett, Phys. Rev. B, 43, 12571 (1991).

(Daw) M. S. Daw, and M. I. Baskes, Phys. Rev. B, 29, 6443 (1984).

4.225 pair_style python command

4.225.1 Syntax

```
pair_style python cutoff
```

cutoff = global cutoff for interactions in python potential classes

4.225.2 Examples

```
pair_style python 2.5
pair_coeff * * py_pot.LJCutMelt lj

pair_style python 10.0
pair_coeff * * py_pot.HarmonicCut A B

pair_style hybrid/overlay coul/long 12.0 python 12.0
pair_coeff * * coul/long
pair_coeff * * python py_pot.LJCutSPCE OW NULL
```

4.225.3 Description

The *python* pair style provides a way to define pairwise additive potential functions as python script code that is loaded into LAMMPS from a python file which must contain specific python class definitions. This allows to rapidly evaluate different potential functions without having to modify and re-compile LAMMPS. Due to python being an interpreted language, however, the performance of this pair style is going to be significantly slower (often between 20x and 100x) than corresponding compiled code. This penalty can be significantly reduced through generating tabulations from the python code through the *pair_write* command, which is supported by this style.

Only a single *pair_coeff* command is used with the *python* pair style which specifies a python class inside a python module or a file that LAMMPS will look up in the current directory, a folder pointed to by the LAMMPS_POTENTIALS environment variable or somewhere in your python path. A single python module can hold multiple python pair class definitions. The class definitions itself have to follow specific rules that are explained below.

Atom types in the python class are specified through symbolic constants, typically strings. These are mapped to LAMMPS atom types by specifying N additional arguments after the class name in the pair_coeff command, where N must be the number of currently defined atom types:

As an example, imagine a file *py_pot.py* has a python potential class names *LJCutMelt* with parameters and potential functions for a two Lennard-Jones atom types labeled as 'LJ1' and 'LJ2'. In your LAMMPS input and you would have defined 3 atom types, out of which the first two are supposed to be using the 'LJ1' parameters and the third the 'LJ2' parameters, then you would use the following pair_coeff command:

```
pair_coeff * * py_pot.LJCutMelt LJ1 LJ1 LJ2
```

The first two arguments **must** be * * so as to span all LAMMPS atom types. The first two LJ1 arguments map LAMMPS atom types 1 and 2 to the LJ1 atom type in the LJCutMelt class of the py_pot.py file. The final LJ2 argument maps LAMMPS atom type 3 to the LJ2 atom type the python file. If a mapping value is specified as NULL, the mapping is not performed, any pair interaction with this atom type will be skipped. This can be used when a *python* potential is used as part of the *hybrid* or *hybrid/overlay* pair style. The NULL values are then placeholders for atom types that will be used with other potentials.

The python potential file has to start with the following code:

```
from __future__ import print_function

class LAMMPSPairPotential(object):
    def __init__(self):
        self.pmap=dict()
        self.units='lj'
```

(continues on next page)

(continued from previous page)

```
  def map_coeff(self,name,ltype):
    self.pmap[ltype]=name
  def check_units(self,units):
    if (units != self.units):
      raise Exception("Conflicting units: %s vs. %s" % (self.units,units))
```

Any classes with definitions of specific potentials have to be derived from this class and should be initialize in a similar fashion to the example given below.

Note

The class constructor has to set up a data structure containing the potential parameters supported by this class. It should also define a variable *self.units* containing a string matching one of the options of LAMMPS' *units* command, which is used to verify, that the potential definition in the python class and in the LAMMPS input match.

Here is an example for a single type Lennard-Jones potential class *LJCutMelt* in reduced units, which defines an atom type *lj* for which the parameters epsilon and sigma are both 1.0:

```
class LJCutMelt(LAMMPSPairPotential):
  def __init__(self):
    super(LJCutMelt,self).__init__()
    # set coeffs: 48*eps*sig**12, 24*eps*sig**6,
    #             4*eps*sig**12,  4*eps*sig**6
    self.units = 'lj'
    self.coeff = {'lj'  : {'lj'  : (48.0,24.0,4.0,4.0)}}
```

The class also has to provide two methods for the computation of the potential energy and forces, which have be named *compute_force*, and *compute_energy*, which both take 3 numerical arguments:

- rsq = the square of the distance between a pair of atoms (float)
- itype = the (numerical) type of the first atom
- jtype = the (numerical) type of the second atom

This functions need to compute the (scaled) force and the energy, respectively, and use the result as return value. The functions need to use the *pmap* dictionary to convert the LAMMPS atom type number to the symbolic value of the internal potential parameter data structure. Following the *LJCutMelt* example, here are the two functions:

```
def compute_force(self,rsq,itype,jtype):
    coeff = self.coeff[self.pmap[itype]][self.pmap[jtype]]
    r2inv  = 1.0/rsq
    r6inv  = r2inv*r2inv*r2inv
    lj1 = coeff[0]
    lj2 = coeff[1]
    return (r6inv * (lj1*r6inv - lj2))*r2inv

 def compute_energy(self,rsq,itype,jtype):
    coeff = self.coeff[self.pmap[itype]][self.pmap[jtype]]
    r2inv  = 1.0/rsq
    r6inv  = r2inv*r2inv*r2inv
    lj3 = coeff[2]
    lj4 = coeff[3]
    return (r6inv * (lj3*r6inv - lj4))
```

Note

for consistency with the C++ pair styles in LAMMPS, the *compute_force* function follows the conventions of the Pair::single() methods and does not return the pairwise force directly, but the force divided by the distance between the two atoms, so this value only needs to be multiplied by delta x, delta y, and delta z to conveniently obtain the three components of the force vector between these two atoms.

Below is a more complex example using *real* units and defines an interaction equivalent to:

```
units real
pair_style harmonic/cut
pair_coeff 1 1 0.2 9.0
pair_coeff 2 2 0.4 9.0
```

This uses the default geometric mixing. The equivalent input with pair style *python* is:

```
units real
pair_style python 10.0
pair_coeff * * py_pot.Harmonic A B
```

Note that while for pair style *harmonic/cut* the cutoff is implicitly set to the minimum of the harmonic potential, for pair style python a global cutoff must be set and it must be equal or larger to the implicit cutoff of the potential in python, which has to explicitly return zero force and energy beyond the cutoff. Also, the mixed parameters have to be explicitly provided. The corresponding python code is:

```
class Harmonic(LAMMPSPairPotential):
    def __init__(self):
        super(Harmonic,self).__init__()
        self.units = 'real'
        # set coeffs: K, r0
        self.coeff = {'A' : {'A' : (0.2,9.0),
                             'B' : (math.sqrt(0.2*0.4),9.0)},
                      'B' : {'A' : (math.sqrt(0.2*0.4),9.0),
                             'B' : (0.4,9.0)}}

    def compute_force(self,rsq,itype,jtype):
        coeff = self.coeff[self.pmap[itype]][self.pmap[jtype]]
        r = math.sqrt(rsq)
        delta = coeff[1]-r
        if (r < coeff[1]):
            return 2.0*delta*coeff[0]/r
        else:
            return 0.0

    def compute_energy(self,rsq,itype,jtype):
        coeff = self.coeff[self.pmap[itype]][self.pmap[jtype]]
        r = math.sqrt(rsq)
        delta = coeff[1]-r
        if (r < coeff[1]):
            return delta*delta*coeff[0]
        else:
            return 0.0
```

Performance Impact

The evaluation of scripted python code will slow down the computation of pairwise interactions quite significantly. However, this performance penalty can be worked around through using the python pair style not for the actual simulation, but to generate tabulated potentials using the *pair_write* command. This will also enable GPU or multi-thread acceleration through the GPU, KOKKOS, or OPENMP package versions of the *table* pair style. Please see below for a LAMMPS input example demonstrating how to build a table file:

```
pair_style python 2.5
pair_coeff * * py_pot.LJCutMelt lj
shell rm -f lj.table
pair_write  1 1 2000 rsq 0.01 2.5 lj.table lj
```

Note that it is strongly recommended to try to **delete** the potential table file before generating it. Since the *pair_write* command will always **append** to a table file, while pair style table will use the **first match**. Thus when changing the potential function in the python class, the table pair style will still read the old variant unless the table file is first deleted.

After switching the pair style to *table*, the potential tables need to be assigned to the LAMMPS atom types like this:

```
pair_style      table linear 2000
pair_coeff      1  1  lj.table lj
```

This can also be done for more complex systems. Please see the *examples/python* folders for a few more examples.

4.225.4 Mixing, shift, table, tail correction, restart, rRESPA info

Mixing of potential parameters has to be handled inside the provided python module. The python pair style simply assumes that force and energy computation can be correctly performed for all pairs of atom types as they are mapped to the atom type labels inside the python potential class.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.225.5 Restrictions

This pair style is part of the PYTHON package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.225.6 Related commands

pair_coeff, *pair_write*, *pair style table*

4.225.7 Default

none

4.226 pair_style quip command

4.226.1 Syntax

```
pair_style quip
```

4.226.2 Examples

```
pair_style      quip
pair_coeff      * * gap_example.xml "Potential xml_label=GAP_2014_5_8_60_17_10_38_466" 14
pair_coeff      * * sw_example.xml "IP SW" 14
```

4.226.3 Description

Style *quip* provides an interface for calling potential routines from the QUIP package. QUIP is built separately, and then linked to LAMMPS. The most recent version of the QUIP package can be downloaded from GitHub: https://github.com/libAtoms/QUIP. The interface is chiefly intended to be used to run Gaussian Approximation Potentials (GAP), which are described in the following publications: *(Bartok et al)* and *(PhD thesis of Bartok)*.

Only a single pair_coeff command is used with the *quip* style that specifies a QUIP potential file containing the parameters of the potential for all needed elements in XML format. This is followed by a QUIP initialization string. Finally, the QUIP elements are mapped to LAMMPS atom types by specifying N atomic numbers, where N is the number of LAMMPS atom types:

- QUIP filename
- QUIP initialization string
- N atomic numbers = mapping of QUIP elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

A QUIP potential is fully specified by the filename which contains the parameters of the potential in XML format, the initialization string, and the map of atomic numbers.

GAP potentials can be obtained from the GAP models and databases page on the libAtoms homepage `https://libatoms.github.io, where the appropriate initialization strings are also advised. The list of atomic numbers must be matched to the LAMMPS atom types specified in the LAMMPS data file or elsewhere.

Two examples input scripts are provided in the examples/PACKAGES/quip directory.

4.226.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.226.5 Restrictions

This pair style is part of the ML-QUIP package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

QUIP potentials are parameterized in electron-volts and Angstroms and therefore should be used with LAMMPS metal *units*.

QUIP potentials are generally not designed to work with the scaling factors set by the *special_bonds* command. The recommended setting in molecular systems is to include all interactions, i.e. to use *special_bonds lj/coul 1.0 1.0 1.0*. Scaling factors > 0.0 will be ignored and treated as 1.0. The only exception to this rule is if you know that your QUIP potential needs to exclude bonded, 1-3, or 1-4 interactions and does not already do this exclusion within QUIP. Then a factor 0.0 needs to be used which will remove such pairs from the neighbor list. This needs to be very carefully tested, because it may remove pairs from the neighbor list that are still required.

4.226.6 Related commands

pair_coeff

(Bartok2010) AP Bartok, MC Payne, R Kondor, and G Csanyi, Physical Review Letters 104, 136403 (2010).

(Bartok_PhD) A Bartok-Partay, PhD Thesis, University of Cambridge, (2010).

4.227 pair_style rann command

4.227.1 Syntax

```
pair_style rann
pair_coeff file Type1_element Type2_element Type3_element...
```

4.227.2 Examples

```
pair_style rann
pair_coeff * * Mg.rann Mg
pair_coeff * * MgAlalloy.rann Mg Mg Al Mg
```

4.227.3 Description

Pair style *rann* computes pairwise interactions for a variety of materials using rapid atomistic neural network (RANN) potentials (*Dickel* , *Nitol*). Neural network potentials work by first generating a series of symmetry functions i.e. structural fingerprints from the neighbor list and then using these values as the input layer of a neural network. There is a single output neuron in the final layer which is the energy. Atomic forces are found by analytical derivatives computed via back-propagation. For alloy systems, each element has a unique network.

4.227.4 Potential file syntax

The RANN potential is defined by a single text file which contains all the fitting parameters for the alloy system. The potential file also defines the active fingerprints, network architecture, activation functions, etc. The potential file is divided into several sections which are identified by one of the following keywords:

- atomtypes
- mass
- fingerprintsperelement
- fingerprints
- fingerprintconstants
- screening (optional)
- networklayers
- layersize
- weight
- bias
- activationfunctions
- calibrationparameters (ignored)

The '#' character is treated as a comment marker, similar to LAMMPS input scripts. Sections are not required to follow a rigid ordering, but do require previous definition of prerequisite information. E.g., fingerprintconstants for a particular fingerprint must follow the fingerprints definition; layersize for a particular layer must follow the declaration of network layers.

atomtypes are defined as follows using element keywords separated by spaces.

```
atomtypes:
Fe Mg Al etc.
```

mass must be specified for each element keyword as follows:

```
mass:Mg:
24.305
mass:Fe:
55.847
mass:Al:
26.982
```

fingerprintsperelement specifies how many fingerprints are active for computing the energy of a given atom. This number must be specified for each element keyword. Active elements for each fingerprint depend upon the type of the central atom and the neighboring atoms. Pairwise fingerprints may be defined for a Mg atom based exclusively on

its Al neighbors, for example. Bond fingerprints may use two neighbor lists of different element types. In computing fingerprintsperelement from all defined fingerprints, only the fingerprints defined for atoms of a particular element should be considered, regardless of the elements used in its neighbor list. In the following code, for example, some fingerprints may compute pairwise fingerprints summing contributions about Fe atoms based on a neighbor list of exclusively Al atoms, but if there are no fingerprints summing contributions of all neighbors about a central Al atom, then fingerprintsperelement of Al is zero:

```
fingerprintsperelement:Mg:
5
fingerprintsperelement:Fe:
2
fingerprintsperelement:Al:
0
```

fingerprints specifies the active fingerprints for a certain element combination. Pair fingerprints are specified for two elements, while bond fingerprints are specified for three elements. Only one fingerprints header should be used for an individual combination of elements. The ordering of the fingerprints in the network input layer is determined by the order of element combinations specified by subsequent *fingerprints* lines, and the order of the fingerprints defined for each element combination. Multiple fingerprints of the same style or different ones may be specified. If the same style and element combination is used for multiple fingerprints, they should have different id numbers. The first element specifies the atoms for which this fingerprint is computed while the other(s) specify which atoms to use in the neighbor lists for the computation. Switching the second and third element type in bond fingerprints has no effect on the computation:

```
fingerprints:Mg_Mg:
radial_0 radialscreened_0 radial_1
fingerprints:Mg_Al_Fe:
bond_0 bondspin_0
fingerprints:Mg_Al:
radial_0 radialscreened_0
```

The following fingerprint styles are currently defined. See the *formulation section* below for their definitions:

- radial
- radialscreened
- radialspin
- radialscreenedspin
- bond
- bondscreened
- bondspin
- bondscreenedspin

fingerprintconstants specifies the meta-parameters for a defined fingerprint. For all radial styles, re, rc, alpha, dr, o, and n must be specified. re should usually be the stable interatomic distance, rc is the cutoff radius, dr is the cutoff smoothing distance, o is the lowest radial power term (which may be negative), and n is the highest power term. The total length of the fingerprint vector is (n-o+1). alpha is a list of decay parameters used for exponential decay of radial contributions. It may be set proportionally to the bulk modulus similarly to MEAM potentials, but other values may provided better fitting in special cases. Bond style fingerprints require specification of re, rc, alphak, dr, k, and m. Here m is the power of the bond cosines and k is the number of decay parameters. Cosine powers go from 0 to m-1 and are each computed for all values of alphak. Thus the total length of the fingerprint vector is m*k.

```
fingerprintconstants:Mg_Mg:radialscreened_0:re:
3.193592
fingerprintconstants:Mg_Mg:radialscreened_0:rc:
6.000000
fingerprintconstants:Mg_Mg:radialscreened_0:alpha:
5.520000 5.520000 5.520000 5.520000 5.520000
fingerprintconstants:Mg_Mg:radialscreened_0:dr:
2.806408
fingerprintconstants:Mg_Mg:radialscreened_0:o:
-1
fingerprintconstants:Mg_Mg:radialscreened_0:n:
3
```

screening specifies the Cmax and Cmin values used in the screening fingerprints. Contributions form neighbors to the fingerprint are omitted if they are blocked by a closer neighbor, and reduced if they are partially blocked. Larger values of Cmin correspond to neighbors being blocked more easily. Cmax cannot be greater than 3, and Cmin cannot be greater than Cmax or less than zero. Screening may be omitted in which case the default values Cmax = 2.8, Cmin = 0.8 are used. Since screening is a bond computation, it is specified separately for each combination of three elements in which the latter two may be interchanged with no effect.

```
screening:Mg_Mg_Mg:Cmax:
2.700000
screening:Mg_Mg_Mg:Cmin:
0.400000
```

networklayers species the size of the neural network for each atom. It counts both the input and output layer and so is 2 + <hidden layers>.

```
networklayers:Mg:
3
```

layersize specifies the length of each layer, including the input layer and output layer. The input layer is layer 0. The size of the input layer size must match the summed length of all the fingerprints for that element, and the output layer size must be 1:

```
layersize:Mg:0:
14
layersize:Mg:1:
20
layersize:Mg:2:
1
```

weight specifies the weight for a given element and layer. Weight cannot be specified for the output layer. The weight of layer i is a *m* x *n* matrix where *m* is the layer size of *i* and *n* is the layer size of *i*+1:

```
weight:Mg:0:
w11 w12 w13 ...
w21 w22 w23 ...
...
```

bias specifies the bias for a given element and layer. Bias cannot be specified for the output layer. The bias of layer i is a nx1 vector where n is the layer size of i+1:

```
bias:Mg:0:
b1
b2
b3
...
```

activationfunctions specifies the activation function for a given element and layer. Activation functions cannot be specified for the output layer:

```
activationfunctions:Mg:0:
sigI
activationfunctions:Mg:1:
linear
```

The following activation styles are currently specified. See the *formulation section* below for their definitions.

- sigI
- linear

calibrationparameters specifies a number of parameters used to calibrate the potential. These are ignored by LAMMPS.

4.227.5 Formulation

In the RANN formulation, the total energy of a system of atoms is given by:

$$E = \sum_{\alpha} E^{\alpha}$$

$$E^{\alpha} = {}^{N}A^{\alpha}$$

$${}^{n+1}A_i^{\alpha} = {}^{n}F\left({}^{n}W_{ij}\ {}^{n}A_j^{\alpha} + {}^{n}B_i\right)$$

$${}^{0}A_i^{\alpha} = \begin{bmatrix} {}^{1}Sf^{\alpha} \\ {}^{2}Sf^{\alpha} \\ \cdots \end{bmatrix}$$

Here E^{α} is the energy of atom α, ${}^{n}F()$, ${}^{n}W_{ij}$ and ${}^{n}B_i$ are the activation function, weight matrix and bias vector of the n-th layer respectively. The inputs to the first layer are a collection of structural fingerprints which are collected and reshaped into a single long vector. The individual fingerprints may be defined in any order and have various shapes and sizes. Multiple fingerprints of the same type and varying parameters may also be defined in the input layer.

Eight types of structural fingerprints are currently defined. In the following, β and γ span the full neighbor list of atom α. δ_i are decay meta-parameters, and r_e is a meta-parameter roughly proportional to the first neighbor distance. r_c and dr are the neighbor cutoff distance and cutoff smoothing distance respectively. $S^{\alpha\beta}$ is the MEAM screening function *(Baskes)*, s_i^{α} and s_i^{β} are the atom spin vectors *(Tranchida)*. $r^{\alpha\beta}$ is the distance from atom α to atom β, and $\theta^{\alpha\beta\gamma}$ is the bond angle:

$$cos\left(\theta^{\alpha\beta\gamma}\right) = \frac{\mathbf{r}^{\alpha\beta} \cdot \mathbf{r}^{\alpha\gamma}}{r^{\alpha\beta} r^{\alpha\gamma}}$$

$S^{\alpha\beta}$ is defined as *(Baskes)*:

$$X^{\gamma\beta} = \left(\frac{r^{\gamma\beta}}{r^{\alpha\beta}}\right)^2$$

$$X^{\alpha\gamma} = \left(\frac{r^{\alpha\gamma}}{r^{\alpha\beta}}\right)^2$$

$$C = \frac{2\left(X^{\alpha\gamma} + X^{\gamma\beta}\right) - \left(X^{\alpha\gamma} - X^{\gamma\beta}\right)^2 - 1}{1 - \left(X^{\alpha\gamma} - X^{\gamma\beta}\right)^2}$$

$$f_c(x) = \begin{cases} 1 & x \geq 1 \\ \left(1 - (1-x)^4\right)^2 & 0 < x < 1 \\ 0 & x \leq 0 \end{cases}$$

$$S^{\alpha\beta\gamma} = f_c\left(\frac{C - C_{min}}{C_{max} - C_{min}}\right)$$

$$S^{\alpha\beta} = \prod_{\gamma} S^{\alpha\beta\gamma}$$

The structural fingerprints are computed as follows:

- **radial**

$${}^{r}Sf_i^{\alpha} = \sum_{\beta}\left(\frac{r^{\alpha\beta}}{r_e}\right)^i e^{-\delta_i \frac{r^{\alpha\beta}}{r_e}} f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right)$$

- **bond**

$${}^{b}Sf_{ij}^{\alpha} = \sum_{\beta}\sum_{\gamma}\left(cos(\theta_{\alpha\beta\gamma})\right)^i e^{-\delta_j \frac{r^{\alpha\beta}}{r_e}} e^{-\delta_j \frac{r^{\alpha\gamma}}{r_e}} f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right) f_c\left(\frac{r_c - r^{\alpha\gamma}}{dr}\right)$$

- **radialscreened**

$${}^{rsc}Sf_i^{\alpha} = \sum_{\beta}\left(\frac{r^{\alpha\beta}}{r_e}\right)^i e^{-\delta_i \frac{r^{\alpha\beta}}{r_e}} S^{\alpha\beta} f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right)$$

- **bondscreened**

$${}^{bsc}Sf_{ij}^{\alpha} = \sum_{\beta}\sum_{\gamma}\left(cos(\theta_{\alpha\beta\gamma})\right)^i e^{-\delta_j \frac{r^{\alpha\beta}}{r_e}} e^{-\delta_j \frac{r^{\alpha\gamma}}{r_e}} S^{\alpha\beta} S^{\alpha\gamma} f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right) f_c\left(\frac{r_c - r^{\alpha\gamma}}{dr}\right)$$

- **radialspin**

$${}^{rsp}Sf_i^{\alpha} = \sum_{\beta}\left(\frac{r^{\alpha\beta}}{r_e}\right)^i e^{-\delta_i \frac{r^{\alpha\beta}}{r_e}} \left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\beta}\right) f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right)$$

- **bondspin**

$${}^{bsp}Sf_{ij}^{\alpha} = \sum_{\beta}\sum_{\gamma}\left(cos(\theta_{\alpha\beta\gamma})\right)^i e^{-\delta_j \frac{r^{\alpha\beta}}{r_e}} e^{-\delta_j \frac{r^{\alpha\gamma}}{r_e}} \left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\beta}\right)\left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\gamma}\right) f_c\left(\frac{r_c - r^{\alpha\beta}}{dr}\right) f_c\left(\frac{r_c - r^{\alpha\gamma}}{dr}\right)$$

- **radialscreenedspin**

$$^{rscsp}Sf_i^{\alpha} = \sum_{\beta} \left(\frac{r^{\alpha\beta}}{r_e} \right)^i e^{-\delta_i \frac{r^{\alpha\beta}}{r_e}} S^{\alpha\beta} \left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\beta} \right) f_c \left(\frac{r_c - r^{\alpha\beta}}{dr} \right)$$

- **bondscreenedspin**

$$^{bscsp}Sf_{ij}^{\alpha} = \sum_{\beta} \sum_{\gamma} \left(cos(\theta_{\alpha\beta\gamma}) \right)^i e^{-\delta_j \frac{r^{\alpha\beta}}{r_e}} e^{-\delta_j \frac{r^{\alpha\gamma}}{r_e}} S^{\alpha\beta} S^{\alpha\gamma} \left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\beta} \right) \left(\mathbf{s}^{\alpha} \cdot \mathbf{s}^{\gamma} \right) f_c \left(\frac{r_c - r^{\alpha\beta}}{dr} \right) f_c \left(\frac{r_c - r^{\alpha\gamma}}{dr} \right)$$

The activation functions are computed as follows:

- **sigI**

$$F^{sigI}(x) = 0.1x + 0.9ln\left(e^x + 1\right)$$

- **linear**

$$F^{linear}(x) = x$$

4.227.6 Restrictions

Pair style *rann* is part of the ML-RANN package. It is only enabled if LAMMPS was built with that package. Additionally, if any spin fingerprint styles are used LAMMPS must be built with the SPIN package as well.

Pair style *rann* does not support computing per-atom stress or using *pair_modify nofdotr*.

4.227.7 Defaults

Cmin = 0.8, Cmax = 2.8.

(Baskes) Baskes, Materials Chemistry and Physics, 50(2), 152-158, (1997).

(Dickel) Dickel, Francis, and Barrett, Computational Materials Science 171 (2020): 109157.

(Nitol) Nitol, Dickel, and Barrett, Computational Materials Science 188 (2021): 110207.

(Tranchida) Tranchida, Plimpton, Thibaudeau and Thompson, Journal of Computational Physics, 372, 406-425, (2018).

4.228 pair_style reaxff command

Accelerator Variants: *reaxff/kk*, *reaxff/omp*

4.228.1 Syntax

```
pair_style reaxff cfile keyword value
```

- cfile = NULL or name of a control file
- zero or more keyword/value pairs may be appended

keyword = checkqeq or lgvdw or safezone or mincap or minhbonds or tabulate or list/blocking
checkqeq value = yes or no = whether or not to require qeq/reaxff or acks2/reaxff fix
enobonds value = yes or no = whether or not to tally energy of atoms with no bonds
lgvdw value = yes or no = whether or not to use a low gradient vdW correction
safezone = factor used for array allocation
mincap = minimum size for array allocation
minhbonds = minimum size use for storing hydrogen bonds
tabulate value = size of interpolation table for Lennard-Jones and Coulomb interactions
list/blocking value = yes or no = whether or not to use "blocking" scheme for bond list build

4.228.2 Examples

```
pair_style reaxff NULL
pair_style reaxff controlfile checkqeq no
pair_style reaxff NULL lgvdw yes
pair_style reaxff NULL safezone 1.6 mincap 100
pair_coeff * * ffield.reax C H O N
```

4.228.3 Description

Pair style *reaxff* computes the ReaxFF potential of van Duin, Goddard and co-workers. ReaxFF uses distance-dependent bond-order functions to represent the contributions of chemical bonding to the potential energy. There is more than one version of ReaxFF. The version implemented in LAMMPS uses the functional forms documented in the supplemental information of the following paper: *(Chenoweth et al., 2008)* and matches the version of the reference ReaxFF implementation from Summer 2010. For more technical details about the implementation of ReaxFF in pair style *reaxff*, see the *(Aktulga)* paper. The *reaxff* style was initially implemented as a stand-alone C code and is now converted to C++ and integrated into LAMMPS as a package.

The *reaxff/kk* style is a Kokkos version of the ReaxFF potential that is derived from the *reaxff* style. The Kokkos version can run on GPUs and can also use OpenMP multithreading. For more information about the Kokkos package, see *Packages details* and *Speed kokkos* doc pages. One important consideration when using the *reaxff/kk* style is the choice of either half or full neighbor lists. This setting can be changed using the Kokkos *package* command.

The *reaxff* style differs from the (obsolete) "pair_style reax" command in the implementation details. The *reax* style was a Fortran library, linked to LAMMPS. The *reax* style has been removed from LAMMPS after the 12 December 2018 version.

LAMMPS provides several different versions of ffield.reax in its potentials dir, each called potentials/ffield.reax.label. These are documented in potentials/README.reax.

The format of these files is identical to that used originally by van Duin. We have tested the accuracy of *pair_style reaxff* potential against the original ReaxFF code for the systems mentioned above. You can use other ffield files for specific chemical systems that may be available elsewhere (but note that their accuracy may not have been tested).

Note

We do not distribute a wide variety of ReaxFF force field files with LAMMPS. Adri van Duin's group at PSU is the central repository for this kind of data as they are continuously deriving and updating parameterizations for different classes of materials. You can submit a contact request at the Materials Computation Center (MCC) website https://www.mri.psu.edu/materials-computation-center/connect-mcc, describing the material(s) you are interested in modeling with ReaxFF. They can tell you what is currently available or what it would take to create a suitable ReaxFF parameterization.

The *cfile* setting can be specified as NULL, in which case default settings are used. A control file can be specified which defines values of control variables. Some control variables are global parameters for the ReaxFF potential. Others define certain performance and output settings. Each line in the control file specifies the value for a control variable. The format of the control file is described below.

Note

The LAMMPS default values for the ReaxFF global parameters correspond to those used by Adri van Duin's stand-alone serial code. If these are changed by setting control variables in the control file, the results from LAMMPS and the serial code will not agree.

Examples using *pair_style reaxff* are provided in the examples/reax directory and its subdirectories.

Use of this pair style requires using an *atom_style* that includes a per-atom charge property *or* using *fix property/atom q*. Charges can be set via *read_data* or *set*. Using an initial charge that is close to the result of charge equilibration will speed up that process.

The ReaxFF parameter files provided were created using a charge equilibration (QEq) model for handling the electrostatic interactions. Therefore, by default, LAMMPS requires that either the *fix qeq/reaxff* or the *fix qeq/shielded* or *fix acks2/reaxff* command be used with *pair_style reaxff* when simulating a ReaxFF model, to equilibrate the charges each timestep.

Using the keyword *checkqeq* with the value *no* turns off the check for the QEq fixes, allowing a simulation to be run without charge equilibration. In this case, the static charges you assign to each atom will be used for computing the electrostatic interactions in the system. See the *fix qeq/reaxff* or *fix qeq/shielded* or *fix acks2/reaxff* command documentation for more details.

Using the optional keyword *lgvdw* with the value *yes* turns on the low-gradient correction of ReaxFF for long-range London Dispersion, as described in the *(Liu)* paper. The bundled force field file *ffield.reax.lg* is designed for this correction, and is trained for several energetic materials (see "Liu"). When using *lgvdw yes*, the recommended value for parameter *thb* is 0.01, which can be set in the control file. Note: Force field files are different for the original or lg corrected pair styles, using the wrong ffield file generates an error.

Using the optional keyword *enobonds* with the value *yes*, the energy of atoms with no bonds (i.e. isolated atoms) is included in the total potential energy and the per-atom energy of that atom. If the value *no* is specified then the energy of atoms with no bonds is set to zero. The latter behavior is usual not desired, as it causes discontinuities in the potential energy when the bonding of an atom drops to zero.

Optional keywords *safezone*, *mincap*, and *minhbonds* are used for allocating reaxff arrays. Increasing these values can avoid memory problems, such as segmentation faults and bondchk failed errors, that could occur under certain conditions. These keywords are not used by the Kokkos version, which instead uses a more robust memory allocation scheme that checks if the sizes of the arrays have been exceeded and automatically allocates more memory.

The keyword *tabulate* controls the size of interpolation table for Lennard-Jones and Coulomb interactions. Tabulation may also be set in the control file (see below). If tabulation is set in both the input script and the control file, the value in the control file will be ignored. A size of 10000 is typically used for the interpolation table. A value of 0 means no tabulation will be used.

The keyword *list/blocking* is only supported by the Kokkos version of ReaxFF and ignored otherwise. Setting the value to *yes* enables the "blocking" scheme (dynamically building interaction lists) for the ReaxFF bond neighbor list. This reduces the number of empty interactions and can improve performance in some cases (e.g. large number of atoms/GPU on AMD hardware). It is also enabled by default when running the CPU with Kokkos.

The thermo variable *evdwl* stores the sum of all the ReaxFF potential energy contributions, with the exception of the Coulombic and charge equilibration contributions which are stored in the thermo variable *ecoul*. The output of these quantities is controlled by the *thermo* command.

This pair style tallies a breakdown of the total ReaxFF potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 14. The 14 values correspond to the following sub-categories (the variable names in italics match those used in the original FORTRAN ReaxFF code):

1. *eb* = bond energy
2. *ea* = atom energy
3. *elp* = lone-pair energy
4. *emol* = molecule energy (always 0.0)
5. *ev* = valence angle energy
6. *epen* = double-bond valence angle penalty
7. *ecoa* = valence angle conjugation energy
8. *ehb* = hydrogen bond energy
9. *et* = torsion energy
10. *eco* = conjugation energy
11. *ew* = van der Waals energy
12. *ep* = Coulomb energy
13. *efi* = electric field energy (always 0.0)
14. *eqeq* = charge equilibration energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute reax all pair reaxff
variable eb      equal c_reax[1]
variable ea      equal c_reax[2]
[...]
variable eqeq    equal c_reax[14]
thermo_style custom step temp epair v_eb v_ea [...] v_eqeq
```

Only a single pair_coeff command is used with the *reaxff* style which specifies a ReaxFF potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N indices = ReaxFF elements

The filename is the ReaxFF potential file.

In the ReaxFF potential file, near the top, after the general parameters, is the atomic parameters section that contains element names, each with a couple dozen numeric parameters. If there are M elements specified in the *ffield* file, think of these as numbered 1 to M. Each of the N indices you specify for the N atom types of LAMMPS atoms must be an integer from 1 to M. Atoms with LAMMPS type 1 will be mapped to whatever element you specify as the first index value, etc. If a mapping value is specified as NULL, the mapping is not performed. This can be used when the *reaxff* style is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

As an example, say your LAMMPS simulation has 4 atom types and the elements are ordered as C, H, O, N in the *ffield* file. If you want the LAMMPS atom type 1 and 2 to be C, type 3 to be N, and type 4 to be H, you would use the following pair_coeff command:

```
pair_coeff * * ffield.reax C C N H
```

4.228.4 Control file

The format of a line in the control file is as follows:

```
variable_name value
```

and it may be followed by an "!" character and a trailing comment.

If the value of a control variable is not specified, then default values are used. What follows is the list of variables along with a brief description of their use and default values.

simulation_name
: Output files produced by *pair_style reaxff* carry this name + extensions specific to their contents. Partial energies are reported with a ".pot" extension, while the trajectory file has ".trj" extension.

tabulate_long_range
: To improve performance, long range interactions can optionally be tabulated (0 means no tabulation). Value of this variable denotes the size of the long range interaction table. The range from 0 to long range cutoff (defined in the *ffield* file) is divided into *tabulate_long_range* points. Then at the start of simulation, we fill in the entries of the long range interaction table by computing the energies and forces resulting from van der Waals and Coulomb interactions between every possible atom type pairs present in the input system. During the simulation we consult to the long range interaction table to estimate the energy and forces between a pair of atoms. Linear interpolation is used for estimation. (default value = 0)

energy_update_freq
: Denotes the frequency (in number of steps) of writes into the partial energies file. (default value = 0)

nbrhood_cutoff
: Denotes the near neighbors cutoff (in Angstroms) regarding the bonded interactions. (default value = 5.0)

hbond_cutoff
: Denotes the cutoff distance (in Angstroms) for hydrogen bond interactions.(default value = 7.5. A value of 0.0 turns off hydrogen bonds)

bond_graph_cutoff
: is the threshold used in determining what is a physical bond, what is not. Bonds and angles reported in the trajectory file rely on this cutoff. (default value = 0.3)

thb_cutoff
: cutoff value for the strength of bonds to be considered in three body interactions. (default value = 0.001)

thb_cutoff_sq
: cutoff value for the strength of bond order products to be considered in three body interactions. (default value = 0.00001)

write_freq
: Frequency of writes into the trajectory file. (default value = 0)

traj_title
: Title of the trajectory - not the name of the trajectory file.

atom_info
: 1 means print only atomic positions + charge (default = 0)

atom_forces
1 adds net forces to atom lines in the trajectory file (default = 0)

atom_velocities
1 adds atomic velocities to atoms line (default = 0)

bond_info
1 prints bonds in the trajectory file (default = 0)

angle_info
1 prints angles in the trajectory file (default = 0)

4.228.5 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.228.6 Restrictions

This pair style is part of the REAXFF package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The ReaxFF potential files provided with LAMMPS in the potentials directory are parameterized for *real units*. You can use the ReaxFF pair style with any LAMMPS units, but you would need to create your own potential file with coefficients listed in the appropriate units if your simulation does not use "real" units.

4.228.7 Related commands

pair_coeff, *fix qeq/reaxff*, *fix acks2/reaxff*, *fix reaxff/bonds*, *fix reaxff/species*, *compute reaxff/atom*

4.228.8 Default

The keyword defaults are checkqeq = yes, enobonds = yes, lgvdw = no, safezone = 1.2, mincap = 50, minhbonds = 25, tabulate = 0, list/blocking = yes on CPU, no on GPU.

(Chenoweth_2008) Chenoweth, van Duin and Goddard, Journal of Physical Chemistry A, 112, 1040-1053 (2008).

(Aktulga) Aktulga, Fogarty, Pandit, Grama, Parallel Computing, 38, 245-259 (2012).

(Liu) L. Liu, Y. Liu, S. V. Zybin, H. Sun and W. A. Goddard, Journal of Physical Chemistry A, 115, 11016-11022 (2011).

4.229 pair_style rebomos command

Accelerator Variants: *rebomos/omp*

4.229.1 Syntax

```
pair_style rebomos
```

- rebomos = name of this pair style

4.229.2 Examples

```
pair_style rebomos
pair_coeff * * ../potentials/MoS.rebomos Mo S
```

Example input scripts available: examples/threebody/

4.229.3 Description

Added in version 17Apr2024.

The *rebomos* pair style computes the interactions between molybdenum and sulfur atoms *(Stewart)* utilizing an adaptive interatomic reactive empirical bond order potential that is similar in form to the AIREBO potential *(Stuart)*. The potential is based on an earlier parameterizations for MoS_2 developed by *(Liang)*.

The REBOMoS potential consists of two terms:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} \left[E_{ij}^{\mathrm{REBO}} + E_{ij}^{\mathrm{LJ}} \right]$$

The E^{REBO} term describes the covalently bonded interactions between Mo and S atoms while the E^{LJ} term describes longer range dispersion forces between layers. A cubic spline function is applied to smoothly switch between covalent bonding at short distances to dispersion interactions at longer distances. This allows the model to capture bond formation and breaking events which may occur between adjacent MoS2 layers, edges, defects, and more.

Only a single pair_coeff command is used with the *rebomos* pair style which specifies an REBOMoS potential file with parameters for Mo and S. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- *N* element names = mapping of REBOMoS elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, if your LAMMPS simulation has three atom types and you want the first two to be Mo, and the third to be S, you would use the following pair_coeff command:

```
pair_coeff * * MoS.rebomos Mo Mo S
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first two Mo arguments map LAMMPS atom types 1 and 2 to the Mo element in the REBOMoS file. The final S argument maps LAMMPS atom type 3 to the S element in the REBOMoS file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *rebomos* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.229.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write their information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair styles can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.229.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair potentials require the *newton* setting to be "on" for pair interactions.

The MoS.rebomos potential file provided with LAMMPS (see the potentials directory) is parameterized for metal *units*. You can use the *rebomos* pair style with any LAMMPS units setting, but you would need to create your own REBOMoS potential file with coefficients listed in the appropriate units.

The pair style provided here **only** supports potential files parameterized for the elements molybdenum and sulfur (designated with "Mo" and "S" in the *pair_coeff* command. Using potential files for other elements will trigger an error.

4.229.6 Related commands

pair_coeff, *pair style rebo*

4.229.7 Default

none

(Steward) Stewart, Spearot, Modelling Simul. Mater. Sci. Eng. 21, 045003, (2013).

(Stuart) Stuart, Tutein, Harrison, J Chem Phys, 112, 6472-6486, (2000).

(Liang) Liang, Phillpot, Sinnott Phys. Rev. B79 245110, (2009), Erratum: Phys. Rev. B85 199903(E), (2012)

4.230 pair_style resquared command

Accelerator Variants: *resquared/gpu*, *resquared/omp*

4.230.1 Syntax

```
pair_style resquared cutoff
```

- cutoff = global cutoff for interactions (distance units)

4.230.2 Examples

```
pair_style resquared 10.0
pair_coeff * * 1.0 1.0 1.7 3.4 3.4 1.0 1.0 1.0
```

4.230.3 Description

Style *resquared* computes the RE-squared anisotropic interaction *(Everaers)*, *(Babadi)* between pairs of ellipsoidal and/or spherical Lennard-Jones particles. For ellipsoidal interactions, the potential considers the ellipsoid as being comprised of small spheres of size σ. LJ particles are a single sphere of size σ. The distinction is made to allow the pair style to make efficient calculations of ellipsoid/solvent interactions.

Details for the equations used are given in the references below and in this supplementary document.

Use of this pair style requires the NVE, NVT, or NPT fixes with the *asphere* extension (e.g. *fix nve/asphere*) in order to integrate particle rotation. Additionally, *atom_style ellipsoid* should be used since it defines the rotational state and the size and shape of each ellipsoidal particle.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A12 = Energy Prefactor/Hamaker constant (energy units)
- σ = atomic interaction diameter (distance units)
- $\varepsilon_{i,a}$ = relative well depth of type I for side-to-side interactions
- $\varepsilon_{i,b}$ = relative well depth of type I for face-to-face interactions
- $\varepsilon_{i,c}$ = relative well depth of type I for end-to-end interactions
- $\varepsilon_{j,a}$ = relative well depth of type J for side-to-side interactions
- $\varepsilon_{j,b}$ = relative well depth of type J for face-to-face interactions
- $\varepsilon_{j,c}$ = relative well depth of type J for end-to-end interactions
- cutoff (distance units)

The parameters used depend on the type of the interacting particles, i.e. ellipsoids or LJ spheres. The type of a particle is determined by the diameters specified for its 3 shape parameters. If all 3 shape parameters = 0.0, then the particle is treated as an LJ sphere. The $\varepsilon_{i,*}$ or $\varepsilon_{j,*}$ parameters are ignored for LJ spheres. If the 3 shape parameters are > 0.0, then the particle is treated as an ellipsoid (even if the 3 parameters are equal to each other).

A12 specifies the energy prefactor which depends on the types of the two interacting particles.

For ellipsoid/ellipsoid interactions, the interaction is computed by the formulas in the supplementary document referenced above. A12 is the Hamaker constant as described in *(Everaers)*. In LJ units:

$$A_{12} = 4\pi^2 \varepsilon_{\mathrm{LJ}} (\rho \sigma^3)^2$$

where ρ gives the number density of the spherical particles composing the ellipsoids and $\varepsilon_{\mathrm{LJ}}$ determines the interaction strength of the spherical particles.

For ellipsoid/LJ sphere interactions, the interaction is also computed by the formulas in the supplementary document referenced above. A12 has a modified form (see here for details):

$$A_{12} = 4\pi^2 \varepsilon_{\mathrm{LJ}} (\rho \sigma^3)$$

For ellipsoid/LJ sphere interactions, a correction to the distance- of-closest approach equation has been implemented to reduce the error from two particles of disparate sizes; see this supplementary document.

For LJ sphere/LJ sphere interactions, the interaction is computed using the standard Lennard-Jones formula, which is much cheaper to compute than the ellipsoidal formulas. A12 is used as epsilon in the standard LJ formula:

$$A_{12} = \varepsilon_{\mathrm{LJ}}$$

and the specified σ is used as the σ in the standard LJ formula.

When one of both of the interacting particles are ellipsoids, then σ specifies the diameter of the continuous distribution of constituent particles within each ellipsoid used to model the RE-squared potential. Note that this is a different meaning for σ than the *pair_style gayberne* potential uses.

The ε_i and ε_j coefficients are defined for atom types, not for pairs of atom types. Thus, in a series of pair_coeff commands, they only need to be specified once for each atom type.

Specifically, if any of $\varepsilon_{i,a}$, $\varepsilon_{i,b}$, $\varepsilon_{i,c}$ are non-zero, the three values are assigned to atom type I. If all the ε_i values are zero, they are ignored. If any of $\varepsilon_{j,a}$, $\varepsilon_{j,b}$, $\varepsilon_{j,c}$ are non-zero, the three values are assigned to atom type J. If all three ε_i values are zero, they are ignored. Thus the typical way to define the ε_i and ε_j coefficients is to list their values in "pair_coeff I J" commands when I = J, but set them to 0.0 when I != J. If you do list them when I != J, you should ensure they are consistent with their values in other pair_coeff commands.

Note that if this potential is being used as a sub-style of *pair_style hybrid*, and there is no "pair_coeff I I" setting made for RE-squared for a particular type I (because I-I interactions are computed by another hybrid pair potential), then you still need to ensure the epsilon a,b,c coefficients are assigned to that type in a "pair_coeff I J" command.

For large uniform molecules it has been shown that the $\varepsilon_{*,*}$ energy parameters are approximately representable in terms of local contact curvatures *(Everaers)*:

$$\varepsilon_a = \sigma \cdot \frac{a}{b \cdot c}; \varepsilon_b = \sigma \cdot \frac{b}{a \cdot c}; \varepsilon_c = \sigma \cdot \frac{c}{a \cdot b}$$

where a, b, and c give the particle diameters.

The last coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.230.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance can be mixed, but only for sphere pairs. The default mix value is *geometric*. See the "pair_modify" command for details. Other type pairs cannot be mixed, due to the different meanings of the energy prefactors used to calculate the interactions and the implicit dependence of the ellipsoid-sphere interaction on the equation for the Hamaker constant presented here. Mixing of sigma and epsilon followed by calculation of the energy prefactors using the equations above is recommended.

This pair style supports the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction, but only for sphere-sphere interactions. There is no shifting performed for ellipsoidal interactions due to the anisotropic dependence of the interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords of the *run_style command*.

4.230.5 Restrictions

This style is part of the ASPHERE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires that atoms be ellipsoids as defined by the *atom_style ellipsoid* command.

Particles acted on by the potential can be finite-size aspherical or spherical particles, or point particles. Spherical particles have all 3 of their shape parameters equal to each other. Point particles have all 3 of their shape parameters equal to 0.0.

The distance-of-closest-approach approximation used by LAMMPS becomes less accurate when high-aspect ratio ellipsoids are used.

4.230.6 Related commands

pair_coeff, *fix nve/asphere*, *compute temp/asphere*, *pair_style gayberne*

4.230.7 Default

none

(Everaers) Everaers and Ejtehadi, Phys Rev E, 67, 041710 (2003).

(Babadi) Babadi, Ejtehadi, Everaers, J Comp Phys, 219, 770-779 (2006).

4.231 pair_style rheo command

4.231.1 Syntax

```
pair_style rheo cutoff keyword values
```

- cutoff = global cutoff for kernel (distance units)
- zero or more keyword/value pairs may be appended to args
- keyword = *rho/damp* or *artificial/visc* or *harmonic/means*

rho/damp args = density damping prefactor ξ
artificial/visc args = artificial viscosity prefactor ζ
harmonic/means args = none

4.231.2 Examples

```
pair_style rheo 3.0 rho/damp 1.0 artificial/visc 2.0
pair_coeff * *
```

4.231.3 Description

Added in version 29Aug2024.

Pair style *rheo* computes pressure and viscous forces between particles in the *rheo package*. If thermal evolution is turned on in *fix rheo*, then the pair style also calculates heat exchanged between particles.

The *artificial/viscosity* keyword is used to specify the magnitude ζ of an optional artificial viscosity contribution to forces. This factor can help stabilize simulations by smoothing out small length scale variations in velocity fields. Artificial viscous forces typically are only exchanged by fluid particles. However, if interfaces are not reconstructed in fix rheo, fluid particles will also exchange artificial viscous forces with solid particles to improve stability.

The *rho/damp* keyword is used to specify the magnitude ξ of an optional pairwise damping term between the density of particles. This factor can help stabilize simulations by smoothing out small length scale variations in density fields. However, in systems that develop a density gradient in equilibrium (e.g. in a hydrostatic column underlying gravity), this option may be inappropriate.

If particles have different viscosities or conductivities, the *harmonic/means* keyword changes how they are averaged before calculating pairwise forces or heat exchanges. By default, an arithmetic averaged is used, however, a harmonic mean may improve stability in systems with multiple fluid phases with large disparities in viscosities.

No coefficients are defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

4.231.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.231.5 Restrictions

This fix is part of the RHEO package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.231.6 Related commands

fix rheo, *fix rheo/pressure*, *fix rheo/thermal*, *fix rheo/viscosity*, *compute rheo/property/atom*

4.231.7 Default

Density damping and artificial viscous forces are not calculated. Arithmetic means are used for mixing particle properties.

4.232 pair_style rheo/solid command

4.232.1 Syntax

```
pair_style rheo/solid
```

4.232.2 Examples

```
pair_style rheo/solid
pair_coeff * * 1.0 1.5 1.0
```

4.232.3 Description

Added in version 29Aug2024.

Style *rheo/solid* is effectively a copy of pair style *bpm/spring* except it only applies forces between solid RHEO particles, determined by checking the status of each pair of neighboring particles before calculating forces.

The style computes pairwise forces with the formula

$$F = k(r - r_c)$$

where k is a stiffness and r_c is the cutoff length. An additional damping force is also applied to interacting particles. The force is proportional to the difference in the normal velocity of particles

$$F_D = -\gamma w(\hat{r} \bullet \vec{v})$$

where γ is the damping strength, $\hat{r}$ is the displacement normal vector, $\vec{v}$ is the velocity difference between the two particles, and w is a smoothing factor. This smoothing factor is constructed such that damping forces go to zero as particles come out of contact to avoid discontinuities. It is given by

$$w = 1.0 - \left(\frac{r}{r_c}\right)^8.$$

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- k (force/distance units)
- r_c (distance units)
- γ (force/velocity units)

4.232.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A coefficient and cutoff distance for this pair style can be mixed. A is always mixed via a *geometric* rule. The cutoff is mixed according to the pair_modify mix value. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option, since the pair interaction goes to 0.0 at the cutoff.

The *pair_modify* table and tail options are not relevant for this pair style.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.232.5 Restrictions

This pair style is part of the RHEO package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.232.6 Related commands

fix rheo, *fix rheo/thermal*, *pair bpm/spring*

4.232.7 Default

none

4.233 pair_style saip/metal command

Accelerator Variant: *saip/metal/opt*

4.233.1 Syntax

```
pair_style [hybrid/overlay ...] saip/metal cutoff tap_flag
```

- cutoff = global cutoff (distance units)
- tap_flag = 0/1 to turn off/on the taper function

4.233.2 Examples

```
pair_style  hybrid/overlay saip/metal 16.0 1
pair_coeff  * * saip/metal CHAu.ILP Au C H

pair_style  hybrid/overlay eam rebo saip/metal 16.0
pair_coeff  1 1 eam  Au_u3.eam  Au NULL NULL
pair_coeff  * * rebo CH.rebo    NULL  C H
pair_coeff  * * saip/metal  CHAu.ILP  Au C H
```

4.233.3 Description

Added in version 17Feb2022.

The *saip/metal* style computes the registry-dependent interlayer potential (ILP) potential for hetero-junctions formed with hexagonal 2D materials and metal surfaces, as described in *(Ouyang6)*.

$$
\begin{aligned}
E =& \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} =& \mathrm{Tap}(r_{ij}) \left\{ e^{-\alpha(r_{ij}/\beta - 1)} \left[\varepsilon + f(\rho_{ij}) + f(\rho_{ji})\right] - \frac{1}{1 + e^{-d\left[(r_{ij}/(s_R \cdot r^{eff})) - 1\right]}} \cdot \frac{C_6}{r_{ij}^6} \right\} \\
\rho_{ij}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_i)^2 \\
\rho_{ji}^2 =& r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_j)^2 \\
f(\rho) =& C e^{-(\rho/\delta)^2} \\
\mathrm{Tap}(r_{ij}) =& 20 \left(\frac{r_{ij}}{R_{cut}}\right)^7 - 70 \left(\frac{r_{ij}}{R_{cut}}\right)^6 + 84 \left(\frac{r_{ij}}{R_{cut}}\right)^5 - 35 \left(\frac{r_{ij}}{R_{cut}}\right)^4 + 1
\end{aligned}
$$

Where $\mathrm{Tap}(r_{ij})$ is the taper function which provides a continuous cutoff (up to third derivative) for interatomic separations larger than r_c *pair_style ilp_graphene_hbn*.

It is important to include all the pairs to build the neighbor list for calculating the normals.

Note

To account for the isotropic nature of the isolated gold atom electron cloud, their corresponding normal vectors (*{bf n}_i*) are assumed to lie along the interatomic vector *{bf r}_ij*. Notably, this assumption is suitable for many bulk material surfaces, for example, for systems possessing s-type valence orbitals or metallic surfaces, whose valence electrons are mostly delocalized, such that their Pauli repulsion with the electrons of adjacent surfaces are isotropic. Caution should be used in the case of very small gold contacts, for example, nano-clusters, where edge effects may become relevant.

The parameter file (e.g. CHAu.ILP), is intended for use with *metal* *units*, with energies in meV. Two additional parameters, *S*, and *rcut* are included in the parameter file. *S* is designed to facilitate scaling of energies. *rcut* is designed to build the neighbor list for calculating the normals for each atom pair.

Note

The parameters presented in the parameter file (e.g. BNCH.ILP), are fitted with taper function by setting the cutoff equal to 16.0 Angstrom. Using different cutoff or taper function should be careful.

This potential must be used in combination with hybrid/overlay. Other interactions can be set to zero using pair_style *none*.

This pair style tallies a breakdown of the total interlayer potential energy into sub-categories, which can be accessed via the *compute pair* command as a vector of values of length 2. The 2 values correspond to the following sub-categories:

1. *E_vdW* = vdW (attractive) energy
2. *E_Rep* = Repulsive energy

To print these quantities to the log file (with descriptive column headings) the following commands could be included in an input script:

```
compute 0 all pair saip/metal
variable Evdw  equal c_0[1]
variable Erep  equal c_0[2]
thermo_style custom step temp epair v_Erep v_Evdw
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.233.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the pair_modify mix, shift, table, and tail options.

This pair style does not write their information to binary restart files, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

4.233.5 Restrictions

This pair style is part of the INTERLAYER package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the newton setting to be *on* for pair interactions.

The CHAu.ILP potential file provided with LAMMPS (see the potentials directory) are parameterized for *metal* units. You can use this potential with any LAMMPS units, but you would need to create your own custom CHAu.ILP potential file with coefficients listed in the appropriate units, if your simulation does not use *metal* units.

4.233.6 Related commands

pair_coeff, *pair_none*, *pair_style hybrid/overlay*, *pair_style drip*, *pair_style ilp_tmd*, *pair_style ilp_graphene_hbn*, *pair_style pair_kolmogorov_crespi_z*, *pair_style pair_kolmogorov_crespi_full*, *pair_style pair_lebedeva_z*, *pair_style pair_coul_shield*.

4.233.7 Default

tap_flag = 1

(Ouyang6) W. Ouyang, O. Hod, and R. Guerra, J. Chem. Theory Comput. 17, 7215 (2021).

4.234 pair_style sdpd/taitwater/isothermal command

4.234.1 Syntax

```
pair_style sdpd/taitwater/isothermal temperature viscosity seed
```

- temperature = temperature of the fluid (temperature units)
- viscosity = dynamic viscosity of the fluid (mass*distance/time units)
- seed = random number generator seed (positive integer, optional)

4.234.2 Examples

```
pair_style sdpd/taitwater/isothermal 300. 1. 28681
pair_coeff * * 1000.0 1430.0 2.4
```

4.234.3 Description

The sdpd/taitwater/isothermal style computes forces between mesoscopic particles according to the Smoothed Dissipative Particle Dynamics model described in this paper by *(Espanol and Revenga)* under the following assumptions:

1. The temperature is constant and uniform.
2. The shear viscosity is constant and uniform.
3. The volume viscosity is negligible before the shear viscosity.
4. The Boltzmann constant is negligible before the heat capacity of a single mesoscopic particle of fluid.

The third assumption is true for water in nearly incompressible flows. The fourth holds true for water for any reasonable size one can imagine for a mesoscopic particle.

The pressure forces between particles will be computed according to Tait's equation of state:

$$p = B\left[(\frac{\rho}{\rho_0})^\gamma - 1\right]$$

where $\gamma = 7$ and $B = c_0^2\rho_0/\gamma$, with ρ_0 being the reference density and c_0 the reference speed of sound.

The laminar viscosity and the random forces will be computed according to formulas described in *(Espanol and Revenga)*.

Warning

Similar to *brownian* and *dpd* styles, the *newton* setting for pairwise interactions needs to be on when running LAMMPS in parallel if you want to ensure linear momentum conservation. Otherwise random forces generated for pairs straddling processor boundary will not be equal and opposite.

Note

The actual random seed used will be a mix of what you specify and other parameters like the MPI ranks. This is to ensure that different MPI tasks have distinct seeds.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- ρ_0 reference density (mass/volume units)
- c_0 reference soundspeed (distance/time units)
- h kernel function cutoff (distance units)

4.234.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.234.5 Restrictions

This pair style is part of the DPD-SMOOTH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.234.6 Related commands

pair coeff, *pair sph/rhosum*, *pair sph/taitwater*

4.234.7 Default

The default seed is 0 (before mixing).

(Espanol and Revenga) Espanol, Revenga, Physical Review E, 67, 026705 (2003).

4.235 pair_style smatb command

4.236 pair_style smatb/single command

4.236.1 Syntax

```
pair_style style args
```

- style = *smatb* or *smatb/single*
- args = none

4.236.2 Examples

```
pair_style smatb
pair_coeff 1 1 2.88 10.35 4.178 0.210 1.818 4.07293506 4.9883063257983666

pair_style smatb/single
pair_coeff 1 1 2.88 10.35 4.178 0.210 1.818 4.07293506 4.9883063257983666
```

4.236.3 Description

Added in version 4May2022.

The *smatb* and *smatb/single* styles compute the Second Moment Approximation to the Tight Binding *(Cyrot)*, *(Gupta)*, *(Rosato)*, given by

$$E_i = \sum_{j,R_{ij} \leq R_c} \alpha(R_{ij}) - \sqrt{\sum_{j,R_{ij} \leq R_c} \Xi^2(R_{ij})}$$

R_{ij} is the distance between the atom i and j. And the two functions $\alpha\,(r)$ and $\Xi\,(r)$ are:

$$\alpha\,(r) = \begin{cases} Ae^{-p\left(\frac{r}{R_0}-1\right)} & r < R_{sc} \\ a_3\,(r-R_c)^3 + a_4\,(r-R_c)^4 + a_5\,(r-R_c)^5 & R_{sc} < r < R_c \end{cases}$$

$$\Xi\,(r) = \begin{cases} \xi e^{-q\left(\frac{r}{R_0}-1\right)} & r < R_{sc} \\ x_3\,(r-R_c)^3 + x_4\,(r-R_c)^4 + x_5\,(r-R_c)^5 & R_{sc} < r < R_c \end{cases}$$

The polynomial coefficients a_3, a_4, a_5, x_3, x_4, x_5 are computed by LAMMPS: the two exponential terms and their first and second derivatives are smoothly reduced to zero, from the inner cutoff R_{sc} to the outer cutoff R_c.

The *smatb/single* style is an optimization when using only a single atom type.

4.236.4 Coefficients

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- R_0 (distance units)
- p (dimensionless)
- q (dimensionless)
- A (energy units)
- ξ (energy units)
- R_{cs} (distance units)
- R_c (distance units)

Note that: R_0 is the nearest neighbor distance, usually coincides with the diameter of the atoms

See the *run_style* command for details.

4.236.5 Mixing info

For atom type pairs I,J and I != J the coefficients are not automatically mixed.

4.236.6 Restrictions

These pair styles are part of the SMTBQ package and are only enabled if LAMMPS is built with that package. See the *Build package* page for more info.

These pair styles require the *newton* setting to be "on" for pair interactions.

4.236.7 Related commands

- *pair_coeff*

4.236.8 Default

none

(Cyrot) Cyrot-Lackmann and Ducastelle, Phys Rev. B, 4, 2406-2412 (1971).

(Gupta) Gupta ,Phys Rev. B, 23, 6265-6270 (1981).

(Rosato) Rosato and Guillope and Legrand, Philosophical Magazine A, 59.2, 321-336 (1989).

4.237 pair_style smd/hertz command

4.237.1 Syntax

```
pair_style smd/hertz scale_factor
```

4.237.2 Examples

```
pair_style smd/hertz 1.0
pair_coeff 1 1 <contact_stiffness>
```

4.237.3 Description

The *smd/hertz* style calculates contact forces between SPH particles belonging to different physical bodies.

The contact forces are calculated using a Hertz potential, which evaluates the overlap between two particles (whose spatial extents are defined via its contact radius). The effect is that a particles cannot penetrate into each other. The parameter <contact_stiffness> has units of pressure and should equal roughly one half of the Young's modulus (or bulk modulus in the case of fluids) of the material model associated with the SPH particles.

The parameter *scale_factor* can be used to scale the particles' contact radii. This can be useful to control how close particles can approach each other. Usually, *scale_factor* =1.0.

4.237.4 Mixing, shift, table, tail correction, restart, rRESPA info

No mixing is performed automatically. Currently, no part of MACHDYN supports restarting nor minimization. rRESPA does not apply to this pair style.

4.237.5 Restrictions

This fix is part of the MACHDYN package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.237.6 Related commands

pair_coeff

4.237.7 Default

none

4.238 pair_style smd/tlsph command

4.238.1 Syntax

```
pair_style smd/tlsph args
```

4.238.2 Examples

```
pair_style smd/tlsph
```

4.238.3 Description

The *smd/tlsph* style computes particle interactions according to continuum mechanics constitutive laws and a Total-Lagrangian Smooth-Particle Hydrodynamics algorithm.

This pair style is invoked with the following command:

```
pair_style smd/tlsph
pair_coeff i j *COMMON rho0 E nu Q1 Q2 hg Cp &
             *END
```

Here, *i* and *j* denote the *LAMMPS* particle types for which this pair style is defined. Note that *i* and *j* must be equal, i.e., no *tlsph* cross interactions between different particle types are allowed. In contrast to the usual *LAMMPS pair coeff* definitions, which are given solely a number of floats and integers, the *tlsph pair coeff* definition is organized using keywords. These keywords mark the beginning of different sets of parameters for particle properties, material constitutive models, and damage models. The *pair coeff* line must be terminated with the **END* keyword. The use the line continuation operator *&* is recommended. A typical invocation of the *tlsph* for a solid body would consist of an equation of state for computing the pressure (the diagonal components of the stress tensor), and a material model to compute shear stresses (the off-diagonal components of the stress tensor). Damage and failure models can also be added.

Please see the SMD user guide for a complete listing of the possible keywords and material models.

4.238.4 Mixing, shift, table, tail correction, restart, rRESPA info

No mixing is performed automatically. Currently, no part of MACHDYN supports restarting nor minimization. rRESPA does not apply to this pair style.

4.238.5 Restrictions

This fix is part of the MACHDYN package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.238.6 Related commands

pair_coeff

4.238.7 Default

none

4.239 pair_style smd/tri_surface command

4.239.1 Syntax

```
pair_style smd/tri_surface scale_factor
```

4.239.2 Examples

```
pair_style smd/tri_surface 1.0
pair_coeff 1 1 <contact_stiffness>
```

4.239.3 Description

The *smd/tri_surface* style calculates contact forces between SPH particles and a rigid wall boundary defined via the *smd/wall_surface* fix.

The contact forces are calculated using a Hertz potential, which evaluates the overlap between a particle (whose spatial extents are defined via its contact radius) and the triangle. The effect is that a particle cannot penetrate into the triangular surface. The parameter <contact_stiffness> has units of pressure and should equal roughly one half of the Young's modulus (or bulk modulus in the case of fluids) of the material model associated with the SPH particle

The parameter *scale_factor* can be used to scale the particles' contact radii. This can be useful to control how close particles can approach the triangulated surface. Usually, *scale_factor* =1.0.

4.239.4 Mixing, shift, table, tail correction, restart, rRESPA info

No mixing is performed automatically. Currently, no part of MACHDYN supports restarting nor minimization. rRESPA does not apply to this pair style.

4.239.5 Restrictions

This fix is part of the MACHDYN package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.239.6 Related commands

pair_coeff

4.239.7 Default

none

4.240 pair_style smd/ulsph command

4.240.1 Syntax

```
pair_style smd/ulsph args
```

- these keywords must be given

keyword = *DENSITY_SUMMATION or *DENSITY_CONTINUITY and *VELOCITY_GRADIENT or *NO_VELOCITY_GRADIENT and *GRADIENT_CORRECTION or *NO_GRADIENT_CORRECTION

4.240.2 Examples

```
pair_style smd/ulsph *DENSITY_CONTINUITY *VELOCITY_GRADIENT *NO_GRADIENT_CORRECTION
```

4.240.3 Description

The *smd/ulsph* style computes particle interactions according to continuum mechanics constitutive laws and an updated Lagrangian Smooth-Particle Hydrodynamics algorithm.

This pair style is invoked similar to the following command:

```
pair_style smd/ulsph *DENSITY_CONTINUITY *VELOCITY_GRADIENT *NO_GRADIENT_CORRECTION
pair_coeff i j *COMMON rho0 c0 Q1 Cp hg &
             *END
```

Here, *i* and *j* denote the *LAMMPS* particle types for which this pair style is defined. Note that *i* and *j* can be different, i.e., *ulsph* cross interactions between different particle types are allowed. However, *i–i* respectively *j–j* pair_coeff lines have to precede a cross interaction. In contrast to the usual *LAMMPS pair coeff* definitions, which are given solely a number of floats and integers, the *ulsph pair coeff* definition is organized using keywords. These keywords mark the beginning of different sets of parameters for particle properties, material constitutive models, and damage models. The *pair coeff* line must be terminated with the **END* keyword. The use the line continuation operator *&* is recommended. A typical invocation of the *ulsph* for a solid body would consist of an equation of state for computing the pressure (the diagonal components of the stress tensor), and a material model to compute shear stresses (the off-diagonal components of the stress tensor).

Note that the use of *GRADIENT_CORRECTION can lead to severe numerical instabilities. For a general fluid simulation, *NO_GRADIENT_CORRECTION is recommended.

Please see the SMD user guide for a complete listing of the possible keywords and material models.

4.240.4 Mixing, shift, table, tail correction, restart, rRESPA info

No mixing is performed automatically. Currently, no part of MACHDYN supports restarting nor minimization. rRESPA does not apply to this pair style.

4.240.5 Restrictions

This fix is part of the MACHDYN package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.240.6 Related commands

pair_coeff

4.240.7 Default

none

4.241 pair_style smtbq command

4.241.1 Syntax

```
pair_style smtbq
```

4.241.2 Examples

```
pair_style smtbq
pair_coeff * * ffield.smtbq.Al2O3 O Al
```

4.241.3 Description

This pair style computes a variable charge SMTB-Q (Second-Moment tight-Binding QEq) potential as described in *SMTB-Q_1* and *SMTB-Q_2*. This potential was first proposed in *SMTB-Q_0*. Briefly, the energy of metallic-oxygen systems is given by three contributions:

$$E_{tot} = E_{ES} + E_{OO} + E_{MO}$$

$$E_{ES} = \sum_i \left[\chi_i^0 Q_i + \frac{1}{2} J_i^0 Q_i^2 + \frac{1}{2} \sum_{j \neq i} J_{ij}(r_{ij}) f_{cut}^{R_{coul}}(r_{ij}) Q_i Q_j \right]$$

$$E_{OO} = \sum_{i,j}^{i,j=O} \left[Cexp(-\frac{r_{ij}}{\rho}) - D f_{cut}^{r_1^{OO} r_2^{OO}}(r_{ij}) exp(Br_{ij}) \right]$$

$$E_{MO} = \sum_i E_{cov}^i + \sum_{j \neq i} A f_{cut}^{r_{c1} r_{c2}}(r_{ij}) exp\left[-p(\frac{r_{ij}}{r_0} - 1)\right]$$

where E_{tot} is the total potential energy of the system, E_{ES} is the electrostatic part of the total energy, E_{OO} is the interaction between oxygen atoms and E_{MO} is a short-range interaction between metal and oxygen atoms. This interactions depend on interatomic distance r_{ij} and/or the charge Q_i of atoms i. Cut-off function enables smooth convergence to zero interaction.

The parameters appearing in the upper expressions are set in the ffield.SMTBQ.Syst file where Syst corresponds to the selected system (e.g. field.SMTBQ.Al2O3). Examples for TiO_2, Al_2O_3 are provided. A single pair_coeff command is used with the SMTBQ styles which provides the path to the potential file with parameters for needed elements. These are mapped to LAMMPS atom types by specifying additional arguments after the potential filename in the pair_coeff command. Note that atom type 1 must always correspond to oxygen atoms. As an example, to simulate a TiO_2 system, atom type 1 has to be oxygen and atom type 2 Ti. The following pair_coeff command should then be used:

```
pair_coeff * * PathToLammps/potentials/ffield.smtbq.TiO2 O Ti
```

The electrostatic part of the energy consists of two components

self-energy of atom i in the form of a second order charge dependent polynomial and a long-range Coulombic electrostatic interaction. The latter uses the wolf summation method described in *Wolf*, spherically truncated at a longer cutoff, R_{coul}. The charge of each ion is modeled by an orbital Slater which depends on the principal quantum number (n) of the outer orbital shared by the ion.

Interaction between oxygen, E_{OO}, consists of two parts, an attractive and a repulsive part. The attractive part is effective only at short range ($< r_2^{OO}$). The attractive contribution was optimized to study surfaces reconstruction (e.g. *SMTB-Q_2* in TiO_2) and is not necessary for oxide bulk modeling. The repulsive part is the Pauli interaction between the electron clouds of oxygen. The Pauli repulsion and the coulombic electrostatic interaction have same cut off value. In the ffield.SMTBQ.Syst, the keyword *'buck'* allows to consider only the repulsive O-O interactions. The keyword *'buckPlusAttr'* allows to consider the repulsive and the attractive O-O interactions.

The short-range interaction between metal-oxygen, E_{MO} is based on the second moment approximation of the density of states with a N-body potential for the band energy term, E_{cov}^i, and a Born-Mayer type repulsive terms as indicated by the keyword *'second_moment'* in the ffield.SMTBQ.Syst. The energy band term is given by:

$$E_{cov}^{i(i=M,O)} = -\left\{ \eta_i (\mu \xi^0)^2 f_{cut}^{r_{c1} r_{c2}}(r_{ij}) \left(\sum_{j(j=O,M)} exp[-2q(\frac{r_{ij}}{r_0} - 1)] \right) \delta Q_i \left(2\frac{n_0}{\eta_i} - \delta Q_i\right) \right\}^{1/2}$$

$$\delta Q_i = |Q_i^F| - |Q_i|$$

where η_i is the stoichiometry of atom i, δQ_i is the charge delocalization of atom i, compared to its formal charge Q_i^F. n_0, the number of hybridized orbitals, is calculated with to the atomic orbitals shared d_i and the stoichiometry η_i. r_{c1} and r_{c2} are the two cutoff radius around the fourth neighbors in the cutoff function.

In the formalism used here, ξ^0 is the energy parameter. ξ^0 is in tight-binding approximation the hopping integral between the hybridized orbitals of the cation and the anion. In the literature we find many ways to write the hopping integral depending on whether one takes the point of view of the anion or cation. These are equivalent vision. The correspondence between the two visions is explained in appendix A of the article in the SrTiO3 *SMTB-Q_3* (parameter β shown in this article is in fact the β_O). To summarize the relationship between the hopping integral ξ^O and the others, we have in an oxide C_nO_m the following relationship:

$$\xi^0 = \frac{\xi_O}{m} = \frac{\xi_C}{n}$$
$$\frac{\beta_O}{\sqrt{m}} = \frac{\beta_C}{\sqrt{n}} = \xi^0 \frac{\sqrt{m}+\sqrt{n}}{2}$$

Thus parameter μ, indicated above, is given by $\mu = \frac{1}{2}(\sqrt{n}+\sqrt{m})$

The potential offers the possibility to consider the polarizability of the electron clouds of oxygen by changing the slater radius of the charge density around the oxygen atoms through the parameters *rBB, rB and rS* in the ffield.SMTBQ.Syst. This change in radius is performed according to the method developed by E. Maras *SMTB-Q_2*. This method needs to determine the number of nearest neighbors around the oxygen. This calculation is based on first (r_{1n}) and second (r_{2n}) distances neighbors.

The SMTB-Q potential is a variable charge potential. The equilibrium charge on each atom is calculated by the electronegativity equalization (QEq) method. See *Rick* for further detail. One can adjust the frequency, the maximum number of iterative loop and the convergence of the equilibrium charge calculation. To obtain the energy conservation in NVE thermodynamic ensemble, we recommend to use a convergence parameter in the interval 10e-5 - 10e-6 eV.

The ffield.SMTBQ.Syst files are provided for few systems. They consist of nine parts and the lines beginning with '#' are comments (note that the number of comment lines matter). The first sections are on the potential parameters and others are on the simulation options and might be modified. Keywords are character type and must be enclosed in quotation marks ('').

1) Number of different element in the oxide:

- N_elem= 2 or 3
- Divider line

2) Atomic parameters

For the anion (oxygen)

- Name of element (char) and stoichiometry in oxide
- Formal charge and mass of element
- Principal quantum number of outer orbital n), electronegativity (χ_i^0) and hardness (J_i^0)
- Ionic radius parameters : max coordination number (*coordBB* = 6 by default), bulk coordination number *(coordB)*, surface coordination number *(coordS)* and *rBB, rB and rS* the slater radius for each coordination number. (**note : If you don't want to change the slater radius, use three identical radius values**)
- Number of orbital shared by the element in the oxide (d_i)
- Divider line

For each cations (metal):

- Name of element (char) and stoichiometry in oxide
- Formal charge and mass of element

- Number of electron in outer orbital *(ne)*, electronegativity (χ_i^0), hardness (J_i^0) and r_{Slater} the slater radius for the cation.
- Number of orbitals shared by the elements in the oxide (d_i)
- Divider line

3) Potential parameters:

- Keyword for element1, element2 and interaction potential ('second_moment' or 'buck' or 'buckPlusAttr') between element 1 and 2. If the potential is 'second_moment', specify 'oxide' or 'metal' for metal-oxygen or metal-metal interactions respectively.
- Potential parameter:
  - If type of potential is 'second_moment' : A (eV), p, ζ^0 (eV) and q, r_{c1}(Å), r_{c2}(Å) and r_0(Å)
  - If type of potential is 'buck' : C (eV) and ρ(Å)
  - If type of potential is 'buckPlusAttr' : C (eV) and ρ(Å) D (eV), B ($Å^{-1}$), r_1^{OO}(Å) and r_2^{OO}(Å)
- Divider line

4) Tables parameters:

- Cutoff radius for the Coulomb interaction (R_{coul})
- Starting radius ($r_{min} = 1,18845$Å) and increments ($dr = 0.001$Å) for creating the potential table.
- Divider line

5) Rick model parameter:

- *Nevery* : parameter to set the frequency of the charge resolution. The charges are evaluated each *Nevery* time steps.
- Max number of iterative loop (*loopmax*) and convergence criterion (*prec*) in eV of the charge resolution
- Divider line

6) Coordination parameter:

- First (r_{1n}) and second (r_{2n}) neighbor distances in angstroms
- Divider line

7) Charge initialization mode:

- Keyword (*QInitMode*) and initial oxygen charge (Q_{init}). If keyword = 'true', all oxygen charges are initially set equal to Q_{init}. The charges on the cations are initially set in order to respect the neutrality of the box. If keyword = 'false', all atom charges are initially set equal to 0 if you use the *create_atoms* command or the charge specified in the file structure using *read_data* command.
- Divider line

8) Mode for the electronegativity equalization (Qeq)

- Keyword (*mode*) followed by:
  - QEqAll (one QEq group) | no parameters
  - QEqAllParallel (several QEq groups) | no parameters
  - Surface | zlim (QEq only for z>zlim)
- Parameter if necessary
- Divider line

9) Verbose

- If you want the code to work in verbose mode or not : 'true' or 'false'
- If you want to print or not in the file 'Energy_component.txt' the three main contributions to the energy of the system according to the description presented above : 'true' or 'false' and N_{Energy}. This option writes to the file every N_{Energy} time steps. If the value is 'false' then N_{Energy} = 0. The file takes into account the possibility to have several QEq groups g then it writes: time step, number of atoms in group g, electrostatic part of energy, E_{ES}, the interaction between oxygen, E_{OO}, and short range metal-oxygen interaction, E_{MO}.
- If you want to print to the file 'Electroneg_component.txt' the electronegativity component ($\frac{\partial E_{tot}}{\partial Q_i}$) or not: 'true' or 'false' and $N_{Electroneg}$. This option writes to the file every $N_{Electroneg}$ time steps. If the value is 'false' then $N_{Electroneg}$ = 0. The file consist of atom number i, atom type (1 for oxygen and # higher than 1 for metal), atom position: x, y and z, atomic charge of atom i, electrostatic part of atom i electronegativity, covalent part of atom i electronegativity, the hopping integral of atom i $(Z\beta^2)_i$ and box electronegativity.

Note

This last option slows down the calculation dramatically. Use only with a single processor simulation.

4.241.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* mix, shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you needs to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.241.5 Restrictions

This pair style is part of the SMTBQ package and is only enabled if LAMMPS is built with that package. See the *Build package* page for more info.

This potential requires using atom type 1 for oxygen and atom type higher than 1 for metal atoms.

This pair style requires the *newton* setting to be "on" for pair interactions.

The SMTB-Q potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*.

4.241.6 Citing this work

Please cite related publication: N. Salles, O. Politano, E. Amzallag and R. Tetot, Comput. Mater. Sci. 111 (2016) 181-189

(SMTB-Q_0) A. Hallil, E. Amzallag, S. Landron, R. Tetot, Surface Science 605 738-745 (2011); R. Tetot, A. Hallil, J. Creuze and I. Braems, EPL, 83 40001 (2008)

(SMTB-Q_1) N. Salles, O. Politano, E. Amzallag, R. Tetot, Comput. Mater. Sci. 111 (2016) 181-189

(SMTB-Q_2) E. Maras, N. Salles, R. Tetot, T. Ala-Nissila, H. Jonsson, J. Phys. Chem. C 2015, 119, 10391-10399

(SMTB-Q_3) R. Tetot, N. Salles, S. Landron, E. Amzallag, Surface Science 616, 19-8722 28 (2013)

(Wolf) D. Wolf, P. Keblinski, S. R. Phillpot, J. Eggebrecht, J Chem Phys, 110, 8254 (1999).

(Rick) S. W. Rick, S. J. Stuart, B. J. Berne, J Chem Phys 101, 6141 (1994).

4.242 pair_style snap command

Accelerator Variants: *snap/intel*, *snap/kk*

4.242.1 Syntax

```
pair_style snap
```

4.242.2 Examples

```
pair_style snap
pair_coeff * * InP.snapcoeff InP.snapparam In In P P
```

4.242.3 Description

Pair style *snap* defines the spectral neighbor analysis potential (SNAP), a machine-learning interatomic potential *(Thompson)*. Like the GAP framework of Bartok et al. *(Bartok2010)*, SNAP uses bispectrum components to characterize the local neighborhood of each atom in a very general way. The mathematical definition of the bispectrum calculation and its derivatives w.r.t. atom positions is identical to that used by *compute snap*, which is used to fit SNAP potentials to *ab initio* energy, force, and stress data. In SNAP, the total energy is decomposed into a sum over atom energies. The energy of atom i is expressed as a weighted sum over bispectrum components.

$$E^i_{SNAP}(B^i_1, ..., B^i_K) = \beta^{\mu_i}_0 + \sum_{k=1}^{K} \beta^{\mu_i}_k B^i_k$$

where B^i_k is the k-th bispectrum component of atom i, and $\beta^{\mu_i}_k$ is the corresponding linear coefficient that depends on μ_i, the SNAP element of atom i. The number of bispectrum components used and their definitions depend on the value of *twojmax* and other parameters defined in the SNAP parameter file described below. The bispectrum calculation is described in more detail in *compute sna/atom*.

Note that unlike for other potentials, cutoffs for SNAP potentials are not set in the pair_style or pair_coeff command; they are specified in the SNAP potential files themselves.

Only a single pair_coeff command is used with the *snap* style which specifies a SNAP coefficient file followed by a SNAP parameter file and then N additional arguments specifying the mapping of SNAP elements to LAMMPS atom types, where N is the number of LAMMPS atom types:

- SNAP coefficient file
- SNAP parameter file
- N element names = mapping of SNAP elements to atom types

As an example, if a LAMMPS indium phosphide simulation has 4 atoms types, with the first two being indium and the third and fourth being phophorous, the pair_coeff command would look like this:

```
pair_coeff * * snap InP.snapcoeff InP.snapparam In In P P
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The two filenames are for the coefficient and parameter files, respectively. The two trailing 'In' arguments map LAMMPS atom types 1 and 2 to the SNAP 'In' element. The two trailing 'P' arguments map LAMMPS atom types 3 and 4 to the SNAP 'P' element.

If a SNAP mapping value is specified as NULL, the mapping is not performed. This can be used when a *snap* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The name of the SNAP coefficient file usually ends in the ".snapcoeff" extension. It may contain coefficients for many SNAP elements. The only requirement is that each of the unique element names appearing in the LAMMPS pair_coeff command appear exactly once in the SNAP coefficient file. It is okay if the SNAP coefficient file contains additional elements not in the pair_coeff command, except when using *chemflag* (see below). The name of the SNAP parameter file usually ends in the ".snapparam" extension. It contains a small number of parameters that define the overall form of the SNAP potential. See the *pair_coeff* page for alternate ways to specify the path for these files.

SNAP potentials are quite commonly combined with one or more other LAMMPS pair styles using the *hybrid/overlay* pair style. As an example, the SNAP tantalum potential provided in the LAMMPS potentials directory combines the *snap* and *zbl* pair styles. It is invoked by the following commands:

```
variable zblcutinner equal 4
variable zblcutouter equal 4.8
variable zblz equal 73
pair_style hybrid/overlay &
zbl ${zblcutinner} ${zblcutouter} snap
pair_coeff * * zbl 0.0
pair_coeff 1 1 zbl ${zblz}
pair_coeff * * snap Ta06A.snapcoeff Ta06A.snapparam Ta
```

It is convenient to keep these commands in a separate file that can be inserted in any LAMMPS input script using the *include* command.

The top of the SNAP coefficient file can contain any number of blank and comment lines (start with #), but follows a strict format after that. The first non-blank non-comment line must contain two integers:

- nelem = Number of elements
- ncoeff = Number of coefficients

This is followed by one block for each of the *nelem* elements. The first line of each block contains three entries:

- Element name (text string)
- R = Element radius (distance units)
- w = Element weight (dimensionless)

This line is followed by *ncoeff* coefficients, one per line.

The SNAP parameter file can contain blank and comment lines (start with #) anywhere. Each non-blank non-comment line must contain one keyword/value pair. The required keywords are *rcutfac* and *twojmax*. Optional keywords are *rfac0*, *rmin0*, *switchflag*, *bzeroflag*, *quadraticflag*, *chemflag*, *bnormflag*, *wselfallflag*, *switchinnerflag*, *sinner*, *dinner*, *chunksize*, and *parallelthresh*.

The default values for these keywords are

- *rfac0* = 0.99363
- *rmin0* = 0.0
- *switchflag* = 1
- *bzeroflag* = 1
- *quadraticflag* = 0
- *chemflag* = 0
- *bnormflag* = 0
- *wselfallflag* = 0
- *switchinnerflag* = 0
- *chunksize* = 32768
- *parallelthresh* = 8192

For detailed definitions of all of these keywords, see the *compute sna/atom* doc page.

If *quadraticflag* is set to 1, then the SNAP energy expression includes additional quadratic terms that have been shown to increase the overall accuracy of the potential without much increase in computational cost *(Wood)*.

$$E^i_{SNAP}(\mathbf{B}^i) = \beta^{\mu_i}_0 + \beta^{\mu_i} \cdot \mathbf{B}_i + \frac{1}{2}\mathbf{B}^t_i \cdot \alpha^{\mu_i} \cdot \mathbf{B}_i$$

where $\mathbf{B}_i$ is the K-vector of bispectrum components, β^{μ_i} is the K-vector of linear coefficients for element μ_i, and α^{μ_i} is the symmetric K by K matrix of quadratic coefficients. The SNAP coefficient file should contain $K(K+1)/2$ additional coefficients in each element block, the upper-triangular elements of α^{μ_i}.

If *chemflag* is set to 1, then the energy expression is written in terms of explicit multi-element bispectrum components indexed on ordered triplets of elements, which has been shown to increase the ability of the SNAP potential to capture energy differences in chemically complex systems, at the expense of a significant increase in computational cost *(Cusentino)*.

$$E^i_{SNAP}(\mathbf{B}^i) = \beta^{\mu_i}_0 + \sum_{\kappa,\lambda,\mu} \beta^{\kappa\lambda\mu}_{\mu_i} \cdot \mathbf{B}^{\kappa\lambda\mu}_i$$

where $\mathbf{B}^{\kappa\lambda\mu}_i$ is the K-vector of bispectrum components for neighbors of elements κ, λ, and μ and $\beta^{\kappa\lambda\mu}_{\mu_i}$ is the corresponding K-vector of linear coefficients for element μ_i. The SNAP coefficient file should contain a total of KN^3_{elem} coefficients in each element block, where N_{elem} is the number of elements in the SNAP coefficient file, which must equal the number of unique elements appearing in the LAMMPS pair_coeff command, to avoid ambiguity in the number of coefficients.

The keyword *switchinnerflag* activates an additional switching function that smoothly turns off contributions to the SNAP potential from neighbor atoms at short separations. If *switchinnerflag* is set to 1 then the additional keywords *sinner* and *dinner* must also be provided. Each of these is followed by *nelements* values, where *nelements* is the number of unique elements appearing in appearing in the LAMMPS pair_coeff command. The element order should correspond to the order in which elements first appear in the pair_coeff command reading from left to right.

The keywords *chunksize* and *parallelthresh* are only applicable when using the pair style *snap* with the KOKKOS package on GPUs and are ignored otherwise. The *chunksize* keyword controls the number of atoms in each pass used to compute the bispectrum components and is used to avoid running out of memory. For example if there are 8192 atoms in the simulation and the *chunksize* is set to 4096, the bispectrum calculation will be broken up into two passes (running on a single GPU). The *parallelthresh* keyword controls a crossover threshold for performing extra parallelism. For small systems, exposing additional parallelism can be beneficial when there is not enough work to fully saturate the GPU threads otherwise. However, the extra parallelism also leads to more divergence and can hurt performance when the system is already large enough to saturate the GPU threads. Extra parallelism will be performed if the *chunksize* (or total number of atoms per GPU) is smaller than *parallelthresh*.

Note

The previously used *diagonalstyle* keyword was removed in 2019, since all known SNAP potentials use the default value of 3.

4.242.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS with user-specifiable parameters as described above. You never need to specify a pair_coeff command with I != J arguments for this style.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.242.5 Restrictions

This style is part of the ML-SNAP package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

The *snap/intel* accelerator variant will *only* be available if LAMMPS is built with Intel *compilers* and for CPUs with AVX-512 support. While the INTEL package in general allows multiple floating point precision modes to be selected, *snap/intel* will currently always use full double precision regardless of the precision mode selected. Additionally, the *intel* variant of snap will **NOT** use multiple threads with OpenMP.

4.242.6 Related commands

compute sna/atom, *compute snad/atom*, *compute snav/atom*, *compute snap*

4.242.7 Default

none

(Thompson) Thompson, Swiler, Trott, Foiles, Tucker, J Comp Phys, 285, 316 (2015).

(Bartok2010) Bartok, Payne, Kondor, Csanyi, Phys Rev Lett, 104, 136403 (2010).

(Wood) Wood and Thompson, J Chem Phys, 148, 241721, (2018)

(Cusentino) Cusentino, Wood, Thompson, J Phys Chem A, 124, 5456, (2020)

4.243 pair_style soft command

Accelerator Variants: *soft/gpu*, *soft/kk*, *soft/omp*

4.243.1 Syntax

```
pair_style soft cutoff
```

- cutoff = global cutoff for soft interactions (distance units)

4.243.2 Examples

```
pair_style soft 1.0
pair_coeff * * 10.0
pair_coeff 1 1 10.0 3.0

pair_style soft 1.0
pair_coeff * * 0.0
variable prefactor equal ramp(0,30)
fix 1 all adapt 1 pair soft a * * v_prefactor
```

4.243.3 Description

Style *soft* computes pairwise interactions with the formula

$$E = A \left[1 + \cos\left(\frac{\pi r}{r_c}\right)\right] \qquad r < r_c$$

It is useful for pushing apart overlapping atoms, since it does not blow up as r goes to 0. A is a prefactor that can be made to vary in time from the start to the end of the run (see discussion below), e.g. to start with a very soft potential and slowly harden the interactions over time. r_c is the cutoff. See the *fix nve/limit* command for another way to push apart overlapping atoms.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- A (energy units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global soft cutoff is used.

Note

The syntax for *pair_coeff* with a single A coeff is different in the current version of LAMMPS than in older versions which took two values, Astart and Astop, to ramp between them. This functionality is now available in a more general form through the *fix adapt* command, as explained below. Note that if you use an old input script and specify Astart and Astop without a cutoff, then LAMMPS will interpret that as A and a cutoff, which is probably not what you want.

The *fix adapt* command can be used to vary A for one or more pair types over the course of a simulation, in which case pair_coeff settings for A must still be specified, but will be overridden. For example these commands will vary the prefactor A for all pairwise interactions from 0.0 at the beginning to 30.0 at the end of a run:

```
variable prefactor equal ramp(0,30)
fix 1 all adapt 1 pair soft a * * v_prefactor
```

Note that a formula defined by an *equal-style variable* can use the current timestep, elapsed time in the current run, elapsed time since the beginning of a series of runs, as well as access other variables.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.243.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A coefficient and cutoff distance for this pair style can be mixed. A is always mixed via a *geometric* rule. The cutoff is mixed according to the pair_modify mix value. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift option, since the pair interaction goes to 0.0 at the cutoff.

The *pair_modify* table and tail options are not relevant for this pair style.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.243.5 Restrictions

none

4.243.6 Related commands

pair_coeff, *fix nve/limit*, *fix adapt*

4.243.7 Default

none

4.244 pair_style sph/heatconduction command

Accelerator Variants: *sph/heatconduction/gpu*

4.244.1 Syntax

```
pair_style sph/heatconduction
```

4.244.2 Examples

```
pair_style sph/heatconduction
pair_coeff * * 1.0 2.4
```

4.244.3 Description

The sph/heatconduction style computes heat transport between SPH particles. The transport model is the diffusion equation for the internal energy.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- D diffusion coefficient (length^2/time units)
- h kernel function cutoff (distance units)

4.244.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.244.5 Restrictions

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.244.6 Related commands

pair_coeff, pair_sph/rhosum

4.244.7 Default

none

4.245 pair_style sph/idealgas command

4.245.1 Syntax

```
pair_style sph/idealgas
```

4.245.2 Examples

```
pair_style sph/idealgas
pair_coeff * * 1.0 2.4
```

4.245.3 Description

The sph/idealgas style computes pressure forces between particles according to the ideal gas equation of state:

$$p = (\gamma - 1)\rho e$$

where $\gamma = 1.4$ is the heat capacity ratio, ρ is the local density, and e is the internal energy per unit mass. This pair style also computes Monaghan's artificial viscosity to prevent particles from interpenetrating *(Monaghan)*.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- ν artificial viscosity (no units)
- h kernel function cutoff (distance units)

4.245.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.245.5 Restrictions

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.245.6 Related commands

pair_coeff, pair_sph/rhosum

4.245.7 Default

none

(Monaghan) Monaghan and Gingold, Journal of Computational Physics, 52, 374-389 (1983).

4.246 pair_style sph/lj command

Accelerator Variants: *sph/lj/gpu*

4.246.1 Syntax

```
pair_style sph/lj
```

4.246.2 Examples

```
pair_style sph/lj
pair_coeff * * 1.0 2.4
```

4.246.3 Description

The sph/lj style computes pressure forces between particles according to the Lennard-Jones equation of state, which is computed according to Ree's 1980 polynomial fit *(Ree)*. The Lennard-Jones parameters epsilon and sigma are set to unity. This pair style also computes Monaghan's artificial viscosity to prevent particles from interpenetrating *(Monaghan)*.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- ν artificial viscosity (no units)
- h kernel function cutoff (distance units)

4.246.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.246.5 Restrictions

As noted above, the Lennard-Jones parameters epsilon and sigma are set to unity.

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.246.6 Related commands

pair_coeff, pair_sph/rhosum

4.246.7 Default

none

(Ree) Ree, Journal of Chemical Physics, 73, 5401 (1980).

(Monaghan) Monaghan and Gingold, Journal of Computational Physics, 52, 374-389 (1983).

4.247 pair_style sph/rhosum command

4.247.1 Syntax

```
pair_style sph/rhosum Nstep
```

- Nstep = timestep interval

4.247.2 Examples

```
pair_style sph/rhosum 10
pair_coeff * * 2.4
```

4.247.3 Description

The sph/rhosum style computes the local particle mass density rho for SPH particles by kernel function interpolation, every Nstep timesteps.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- h (distance units)

4.247.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.247.5 Restrictions

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.247.6 Related commands

pair_coeff, pair_sph/taitwater

4.247.7 Default

none

4.248 pair_style sph/taitwater command

Accelerator Variants: *sph/taitwater/gpu*

4.248.1 Syntax

```
pair_style sph/taitwater
```

4.248.2 Examples

```
pair_style sph/taitwater
pair_coeff * * 1000.0 1430.0 1.0 2.4
```

4.248.3 Description

The sph/taitwater style computes pressure forces between SPH particles according to Tait's equation of state:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right]$$

where $\gamma = 7$ and $B = c_0^2\rho_0/\gamma$, with ρ_0 being the reference density and c_0 the reference speed of sound.

This pair style also computes Monaghan's artificial viscosity to prevent particles from interpenetrating *(Monaghan)*.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- ρ_0 reference density (mass/volume units)
- c_0 reference soundspeed (distance/time units)
- ν artificial viscosity (no units)
- h kernel function cutoff (distance units)

4.248.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.248.5 Restrictions

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.248.6 Related commands

pair_coeff, pair_sph/rhosum

4.248.7 Default

none

(Monaghan) Monaghan and Gingold, Journal of Computational Physics, 52, 374-389 (1983).

4.249 pair_style sph/taitwater/morris command

4.249.1 Syntax

```
pair_style sph/taitwater/morris
```

4.249.2 Examples

```
pair_style sph/taitwater/morris
pair_coeff * * 1000.0 1430.0 1.0 2.4
```

4.249.3 Description

The sph/taitwater/morris style computes pressure forces between SPH particles according to Tait's equation of state:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right]$$

where $\gamma = 7$ and $B = c_0^2\rho_0/\gamma$, with ρ_0 being the reference density and c_0 the reference speed of sound.

This pair style also computes laminar viscosity *(Morris)*.

See this PDF guide to using SPH in LAMMPS.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- ρ_0 reference density (mass/volume units)
- c_0 reference soundspeed (distance/time units)
- ν dynamic viscosity (mass*distance/time units)
- h kernel function cutoff (distance units)

4.249.4 Mixing, shift, table, tail correction, restart, rRESPA info

This style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

This style does not support the *pair_modify* shift, table, and tail options.

This style does not write information to *binary restart files*. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.249.5 Restrictions

This pair style is part of the SPH package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.249.6 Related commands

pair_coeff, pair_sph/rhosum

4.249.7 Default

none

(Morris) Morris, Fox, Zhu, J Comp Physics, 136, 214-226 (1997).

4.250 pair_style lj/spica command

Accelerator Variants: *lj/spica/gpu*, *lj/spica/kk*, *lj/spica/omp*

4.251 pair_style lj/spica/coul/long command

Accelerator Variants: *lj/spica/coul/long/gpu*, *lj/spica/coul/long/omp*, *lj/spica/coul/long/kk*

4.252 pair_style lj/spica/coul/msm command

Accelerator Variants: *lj/spica/coul/msm/omp*

4.252.1 Syntax

```
pair_style style args
```

- style = *lj/spica* or *lj/spica/coul/long*
- args = list of arguments for a particular style

lj/spica args = cutoff
  cutoff = global cutoff for Lennard Jones interactions (distance units)
lj/spica/coul/long args = cutoff (cutoff2)
  cutoff = global cutoff for LJ (and Coulombic if only 1 arg) (distance units)
  cutoff2 = global cutoff for Coulombic (optional) (distance units)

4.252.2 Examples

```
pair_style lj/spica 2.5
pair_coeff 1 1 lj12_6 1 1.1 2.8

pair_style lj/spica/coul/long 10.0
pair_style lj/spica/coul/long 10.0 12.0
pair_coeff 1 1 lj9_6 100.0 3.5 12.0

pair_style lj/spica/coul/msm 10.0
pair_style lj/spica/coul/msm 10.0 12.0
pair_coeff 1 1 lj9_6 100.0 3.5 12.0
```

4.252.3 Description

The *lj/spica* styles compute a 9/6, 12/4, 12/5, or 12/6 Lennard-Jones potential, given by

$$E = \frac{27}{4}\varepsilon\left[\left(\frac{\sigma}{r}\right)^9 - \left(\frac{\sigma}{r}\right)^6\right] \qquad r < r_c$$

$$E = \frac{3\sqrt{3}}{2}\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^4\right] \qquad r < r_c$$

$$E = \frac{12}{7}\left(\frac{12}{5}\right)^{\left(\frac{5}{7}\right)}\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^5\right] \qquad r < r_c$$

$$E = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6\right] \qquad r < r_c$$

as required for the SPICA (formerly called SDK) and the pSPICA Coarse-grained MD parameterization discussed in *(Shinoda)*, *(DeVane)*, *(Seo)*, and *(Miyazaki)*. r_c is the cutoff. Summary information on these force fields can be found at https://www.spica-ff.org

Style *lj/spica/coul/long* computes the adds Coulombic interactions with an additional damping factor applied so it can be used in conjunction with the *kspace_style* command and its *ewald* or *pppm* or *pppm/cg* option. The Coulombic cutoff specified for this style means that pairwise interactions within this distance are computed directly; interactions outside that distance are computed in reciprocal space.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cg_type (lj9_6, lj12_4, lj12_5, or lj12_6)
- epsilon (energy units)
- sigma (distance units)
- cutoff1 (distance units)

Note that sigma is defined in the LJ formula as the zero-crossing distance for the potential, not as the energy minimum. The prefactors are chosen so that the potential minimum is at -epsilon.

The latter 2 coefficients are optional. If not specified, the global LJ and Coulombic cutoffs specified in the pair_style command are used. If only one cutoff is specified, it is used as the cutoff for both LJ and Coulombic interactions for this type pair. If both coefficients are specified, they are used as the LJ and Coulombic cutoffs for this type pair.

For *lj/spica/coul/long* and *lj/spica/coul/msm* only the LJ cutoff can be specified since a Coulombic cutoff cannot be specified for an individual I,J type pair. All type pairs use the same global Coulombic cutoff specified in the pair_style command.

The original implementation of the above styles are style *lj/sdk*, *lj/sdk/coul/long*, and *lj/sdk/coul/msm*, and available for backward compatibility.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.252.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of the lj/spica pair styles *cannot* be mixed, since different pairs may have different exponents. So all parameters for all pairs have to be specified explicitly through the "pair_coeff" command. Defining then in a data file is also not supported, due to limitations of that file format.

All of the lj/spica pair styles support the *pair_modify* shift option for the energy of the Lennard-Jones portion of the pair interaction.

The *lj/spica/coul/long* pair styles support the *pair_modify* table option since they can tabulate the short-range portion of the long-range Coulombic interaction.

All of the lj/spica pair styles write their information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The lj/spica and lj/cut/coul/long pair styles do not support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command.

4.252.5 Restrictions

All of the lj/spica pair styles are part of the CG-SPICA package. The *lj/spica/coul/long* style also requires the KSPACE package to be built (which is enabled by default). They are only enabled if LAMMPS was built with that package. See the *Build package* doc page for more info.

4.252.6 Related commands

pair_coeff , *angle_style spica*

4.252.7 Default

none

(Shinoda) Shinoda, DeVane, Klein, Mol Sim, 33, 27-36 (2007).

(DeVane) Shinoda, DeVane, Klein, Soft Matter, 4, 2453-2462 (2008).

(Seo) Seo, Shinoda, J Chem Theory Comput, 15, 762-774 (2019).

(Miyazaki) Miyazaki, Okazaki, Shinoda, J Chem Theory Comput, 16, 782-793 (2020).

4.253 pair_style spin/dipole/cut command

4.254 pair_style spin/dipole/long command

4.254.1 Syntax

```
pair_style spin/dipole/cut cutoff
pair_style spin/dipole/long cutoff
```

- cutoff = global cutoff for magnetic dipole energy and forces (optional) (distance units)

4.254.2 Examples

```
pair_style spin/dipole/cut 10.0
pair_coeff * * 10.0
pair_coeff 2 3 8.0

pair_style spin/dipole/long 9.0
pair_coeff * * 10.0
pair_coeff 2 3 6.0
```

4.254.3 Description

Style *spin/dipole/cut* computes a short-range dipole-dipole interaction between pairs of magnetic particles that each have a magnetic spin. The magnetic dipole-dipole interactions are computed by the following formulas for the magnetic energy, magnetic precession vector omega and mechanical force between particles I and J.

$$\mathcal{H}_{\text{long}} = -\frac{\mu_0 \left(\mu_B\right)^2}{4\pi} \sum_{i,j,i\neq j}^{N} \frac{g_i g_j}{r_{ij}^3} \left(3 \left(\vec{e}_{ij} \cdot \vec{s}_i\right) \left(\vec{e}_{ij} \cdot \vec{s}_j\right) - \vec{s}_i \cdot \vec{s}_j \right)$$

$$\omega_i = \frac{\mu_0 (\mu_B)^2}{4\pi\hbar} \sum_j \frac{g_i g_j}{r_{ij}^3} \left(3 \left(\vec{e}_{ij} \cdot \vec{s}_j\right) \vec{e}_{ij} - \vec{s}_j \right)$$

$$\mathbf{F}_i = \frac{3\,\mu_0 (\mu_B)^2}{4\pi} \sum_j \frac{g_i g_j}{r_{ij}^4} \left[\left((\vec{s}_i \cdot \vec{s}_j) - 5 (\vec{e}_{ij} \cdot \vec{s}_i)(\vec{e}_{ij} \cdot \vec{s}_j) \right) \vec{e}_{ij} + \left((\vec{e}_{ij} \cdot \vec{s}_i) \vec{s}_j + (\vec{e}_{ij} \cdot \vec{s}_j) \vec{s}_i \right) \right]$$

where $\vec{s}_i$ and $\vec{s}_j$ are the spin on two magnetic particles, r is their separation distance, and the vector $\vec{e}_{ij} = \frac{r_i - r_j}{|r_i - r_j|}$ is the direction vector between the two particles.

Style *spin/dipole/long* computes long-range magnetic dipole-dipole interaction. A *kspace_style* must be defined to use this pair style. Currently, *kspace_style ewald/dipole/spin* and *kspace_style pppm/dipole/spin* support long-range magnetic dipole-dipole interactions.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

4.254.4 Restrictions

The *spin/dipole/cut* and *spin/dipole/long* styles are part of the SPIN package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Using dipole/spin pair styles with *electron units* is not currently supported.

4.254.5 Related commands

pair_coeff , *kspace_style fix nve/spin*

4.254.6 Default

none

4.255 pair_style spin/dmi command

4.255.1 Syntax

```
pair_style spin/dmi cutoff
```

- cutoff = global cutoff pair (distance in metal units)

4.255.2 Examples

```
pair_style spin/dmi 4.0
pair_coeff * * dmi 2.6 0.001 1.0 0.0 0.0
pair_coeff 1 2 dmi 4.0 0.00109 0.0 0.0 1.0
```

4.255.3 Description

Style *spin/dmi* computes the Dzyaloshinskii-Moriya (DM) interaction between pairs of magnetic spins. According to the expression reported in *(Rohart)*, one has the following DM energy:

$$\mathbf{H}_{dm} = \sum_{i,j=1,i\neq j}^{N} \left(\vec{e}_{ij} \times \vec{D}\right) \cdot \left(\vec{s}_i \times \vec{s}_j\right),$$

where $\vec{s}_i$ and $\vec{s}_j$ are two neighboring magnetic spins of two particles, $\vec{e}_i j = \frac{r_i - r_j}{|r_i - r_j|}$ is the unit vector between sites i and j, and $\vec{D}$ is the DM vector defining the intensity (in eV) and the direction of the interaction.

In *(Rohart)*, $\vec{D}$ is defined as the direction normal to the film oriented from the high spin-orbit layer to the magnetic ultra-thin film.

The application of a spin-lattice Poisson bracket to this energy (as described in *(Tranchida)*) allows to derive a magnetic torque omega, and a mechanical force F (for spin-lattice calculations only) for each magnetic particle i:

$$\vec{\omega}_i = -\frac{1}{\hbar} \sum_{j}^{Neighb} \vec{s}_j \times \left(\vec{e}_{ij} \times \vec{D}\right) \quad \text{and} \quad \vec{F}_i = -\sum_{j}^{Neighb} \frac{1}{r_{ij}} \vec{D} \times \left(\vec{s}_i \times \vec{s}_j\right)$$

More details about the derivation of these torques/forces are reported in *(Tranchida)*.

For the *spin/dmi* pair style, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, and set in the following order:

- rc (distance units)
- |D| (energy units)
- Dx, Dy, Dz (direction of D)

Note that rc is the radius cutoff of the considered DM interaction, |D| is the norm of the DM vector (in eV), and Dx, Dy and Dz define its direction.

None of those coefficients is optional. If not specified, the *spin/dmi* pair style cannot be used.

4.255.4 Restrictions

All the *pair/spin* styles are part of the SPIN package. These styles are only enabled if LAMMPS was built with this package, and if the atom_style "spin" was declared. See the *Build package* page for more info.

4.255.5 Related commands

atom_style spin, *pair_coeff*, *pair_eam*,

4.255.6 Default

none

(Rohart) Rohart and Thiaville, Physical Review B, 88(18), 184422. (2013).

(Tranchida) Tranchida, Plimpton, Thibaudeau and Thompson, Journal of Computational Physics, 372, 406-425, (2018).

4.256 pair_style spin/exchange command

4.257 pair_style spin/exchange/biquadratic command

4.257.1 Syntax

```
pair_style spin/exchange cutoff
pair_style spin/exchange/biquadratic cutoff
```

- cutoff = global cutoff pair (distance in metal units)

4.257.2 Examples

```
pair_style spin/exchange 4.0
pair_coeff * * exchange 4.0 0.0446928 0.003496 1.4885
pair_coeff 1 2 exchange 6.0 -0.01575 0.0 1.965 offset yes

pair_style spin/exchange/biquadratic 4.0
pair_coeff * * biquadratic 4.0 0.05 0.03 1.48 0.05 0.03 1.48 offset no
pair_coeff 1 2 biquadratic 6.0 -0.01 0.0 1.9 0.0 0.1 19
```

4.257.3 Description

Style *spin/exchange* computes the exchange interaction between pairs of magnetic spins:

$$H_{ex} = -\sum_{i,j}^{N} J_{ij}(r_{ij})\,\vec{s}_i \cdot \vec{s}_j$$

where $\vec{s}_i$ and $\vec{s}_j$ are two unit vectors representing the magnetic spins of two particles (usually atoms), and $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the inter-atomic distance between those two particles. The summation is over pairs of nearest neighbors. $J(r_{ij})$ is a function defining the intensity and the sign of the exchange interaction for different neighboring shells.

Style *spin/exchange/biquadratic* computes a biquadratic exchange interaction between pairs of magnetic spins:

$$H_{bi} = -\sum_{i,j}^{N} J_{ij}\left(r_{ij}\right)\ \vec{s}_i \cdot \vec{s}_j - \sum_{i,j}^{N} K_{ij}\left(r_{ij}\right)\ \left(\vec{s}_i \cdot \vec{s}_j\right)^2$$

where $\vec{s}_i$, $\vec{s}_j$, r_{ij} and $J(r_{ij})$ have the same definitions as above, and $K(r_{ij})$ is a second function, defining the intensity and the sign of the biquadratic term.

The interatomic dependence of $J(r_{ij})$ and $K(r_{ij})$ in both interactions above is defined by the following function:

$$f\left(r_{ij}\right) = 4a\left(\frac{r_{ij}}{d}\right)^2\left(1 - b\left(\frac{r_{ij}}{d}\right)^2\right)e^{-\left(\frac{r_{ij}}{d}\right)^2}\Theta(R_c - r_{ij})$$

where a, b and d are the three constant coefficients defined in the associated "pair_coeff" command, and R_c is the radius cutoff associated to the pair interaction (see below for more explanations).

The coefficients a, b, and d need to be fitted so that the function above matches with the value of the exchange interaction for the N neighbor shells taken into account. Examples and more explanations about this function and its parameterization are reported in *(Tranchida)*.

When a *spin/exchange/biquadratic* pair style is defined, six coefficients (three for $J(r_{ij})$, and three for $K(r_{ij})$) have to be fitted.

From this exchange interaction, each spin i will be submitted to a magnetic torque $\vec{\omega}_i$, and its associated atom can be submitted to a force $\vec{F}_i$ for spin-lattice calculations (see *fix nve/spin*), such as:

$$\vec{\omega}_i = \frac{1}{\hbar}\sum_j^{Neighb} J\left(r_{ij}\right)\,\vec{s}_j \;\text{ and }\; \vec{F}_i = \sum_j^{Neighb}\frac{\partial J\left(r_{ij}\right)}{\partial r_{ij}}\left(\vec{s}_i\cdot\vec{s}_j\right)\vec{e}_{ij}$$

with $\hbar$ the Planck constant (in metal units), and $\vec{e}_{ij} = \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|}$ the unit vector between sites i and j. Equivalent forces and magnetic torques are generated for the biquadratic term when a *spin/exchange/biquadratic* pair style is defined.

More details about the derivation of these torques/forces are reported in *(Tranchida)*.

For the *spin/exchange* and *spin/exchange/biquadratic* pair styles, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, and set in the following order:

- R_c (distance units)
- a (energy units)
- b (adim parameter)
- d (distance units)

for the *spin/exchange* pair style, and:

- R_c (distance units)
- a_j (energy units)
- b_j (adim parameter)
- d_j (distance units)
- a_k (energy units)
- b_k (adim parameter)
- d_k (distance units)

for the *spin/exchange/biquadratic* pair style.

Note that R_c is the radius cutoff of the considered exchange interaction, and a, b and d are the three coefficients performing the parameterization of the function $J(r_{ij})$ defined above (in the *biquadratic* style, a_j, b_j, d_j and a_k, b_k, d_k are the coefficients of $J(r_{ij})$ and $K(r_{ij})$ respectively).

None of those coefficients is optional. If not specified, the *spin/exchange* pair style cannot be used.

Offsetting magnetic forces and energies:

For spin-lattice simulation, it can be useful to offset the mechanical forces and energies generated by the exchange interaction. The *offset* keyword allows to apply this offset. By setting *offset* to *yes*, the energy definitions above are replaced by:

$$H_{ex} = -\sum_{i,j}^{N} J_{ij}(r_{ij}) \left[\vec{s}_i \cdot \vec{s}_j - 1\right]$$

for the *spin/exchange* pair style, and:

$$H_{bi} = -\sum_{i,j}^{N} J_{ij}\left(r_{ij}\right) \left[\vec{s}_i \cdot \vec{s}_j - 1\right] - \sum_{i,j}^{N} K_{ij}\left(r_{ij}\right) \left[\left(\vec{s}_i \cdot \vec{s}_j\right)^2 - 1\right]$$

for the *spin/exchange/biquadratic* pair style.

Note that this offset only affects the calculation of the energy and mechanical forces. It does not modify the calculation of the precession vectors (and thus does no impact the purely magnetic properties). This ensures that when all spins are aligned, the magnetic energy and the associated mechanical forces (and thus the pressure generated by the magnetic potential) are null.

Note

This offset term can be very important when calculations such as equations of state (energy vs volume, or energy vs pressure) are being performed. Indeed, setting the *offset* term ensures that at the ground state of the crystal and at the equilibrium magnetic configuration (typically ferromagnetic), the pressure is null, as expected. Otherwise, magnetic forces could generate a residual pressure.

When the *offset* option is set to *no*, no offset is applied (also corresponding to the default option).

4.257.4 Restrictions

All the *pair/spin* styles are part of the SPIN package. These styles are only enabled if LAMMPS was built with this package, and if the atom_style "spin" was declared. See the *Build package* page for more info.

4.257.5 Related commands

atom_style spin, *pair_coeff* , *pair_eam*,

4.257.6 Default

The default *offset* keyword value is *no*.

(Tranchida) Tranchida, Plimpton, Thibaudeau and Thompson, Journal of Computational Physics, 372, 406-425, (2018).

4.258 pair_style spin/magelec command

4.258.1 Syntax

```
pair_style spin/magelec cutoff
```

- cutoff = global cutoff pair (distance in metal units)

4.258.2 Examples

```
pair_style spin/magelec 4.5
pair_coeff * * magelec 4.5 0.00109 1.0 1.0 1.0
```

4.258.3 Description

Style *spin/me* computes a magneto-electric interaction between pairs of magnetic spins. According to the derivation reported in *(Katsura)*, this interaction is defined as:

$$\vec{\omega}_i = -\frac{1}{\hbar} \sum_j^{Neighb} \vec{s}_j \times \vec{D}(r_{ij})$$

$$\vec{F}_i = -\sum_j^{Neighb} \frac{\partial D(r_{ij})}{\partial r_{ij}} \left(\vec{s}_i \times \vec{s}_j\right) \cdot \vec{r}_{ij}$$

where $\vec{s}_i$ and $\vec{s}_j$ are neighboring magnetic spins of two particles.

From this magneto-electric interaction, each spin i will be submitted to a magnetic torque omega, and its associated atom can be submitted to a force F for spin-lattice calculations (see *fix nve/spin*), such as:

$$\vec{F}^i = -\sum_j^{Neighbor} \left(\vec{s}_i \times \vec{s}_j\right) \times \vec{E}$$

$$\vec{\omega}^i = -\frac{1}{\hbar} \sum_j^{Neighbor} \vec{s}_j \times \left(\vec{E} \times r_{ij}\right)$$

with h the Planck constant (in metal units) and $\vec{E}$ an electric polarization vector. The norm and direction of E are giving the intensity and the direction of a screened dielectric atomic polarization (in eV).

More details about the derivation of these torques/forces are reported in *(Tranchida)*.

4.258.4 Restrictions

All the *pair/spin* styles are part of the SPIN package. These styles are only enabled if LAMMPS was built with this package, and if the atom_style "spin" was declared. See the *Build package* page for more info.

4.258.5 Related commands

atom_style spin, *pair_coeff*, *pair_style spin/exchange*, *pair_eam*,

4.258.6 Default

none

(Katsura) H. Katsura, N. Nagaosa, A.V. Balatsky. Phys. Rev. Lett., 95(5), 057205. (2005)

(Tranchida) Tranchida, Plimpton, Thibaudeau, and Thompson, Journal of Computational Physics, 372, 406-425, (2018).

4.259 pair_style spin/neel command

4.259.1 Syntax

```
pair_style spin/neel cutoff
```

- cutoff = global cutoff pair (distance in metal units)

4.259.2 Examples

```
pair_style spin/neel 4.0
pair_coeff * * neel 4.0 0.0048 0.234 1.168 2.6905 0.705 0.652
pair_coeff 1 2 neel 4.0 0.0048 0.234 1.168 0.0 0.0 1.0
```

4.259.3 Description

Style *spin/neel* computes the Neel pair anisotropy model between pairs of magnetic spins:

$$\mathcal{H}_{N\acute{e}el} = -\sum_{i,j=1,i\neq j}^{N} g_1(r_{ij})\left((\mathbf{e}_{ij}\cdot\mathbf{s}_i)(\mathbf{e}_{ij}\cdot\mathbf{s}_j)-\frac{\mathbf{s}_i\cdot\mathbf{s}_j}{3}\right)+q_1(r_{ij})\left((\mathbf{e}_{ij}\cdot\mathbf{s}_i)^2-\frac{\mathbf{s}_i\cdot\mathbf{s}_j}{3}\right)\left((\mathbf{e}_{ij}\cdot\mathbf{s}_i)^2-\frac{\mathbf{s}_i\cdot\mathbf{s}_j}{3}\right)+q_2(r_{ij})\Big((\mathbf{e}_{ij}\cdot\mathbf{s}_i)(\mathbf{e}_{ij}\cdot\mathbf{s}_j)^3+(\mathbf{e}_{ij}$$

where $\mathbf{s}_i$ and $\mathbf{s}_j$ are two neighboring magnetic spins of two particles, $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the inter-atomic distance between the two particles, $\mathbf{e}_{ij} = \frac{\mathbf{r}_i-\mathbf{r}_j}{|\mathbf{r}_i-\mathbf{r}_j|}$ is their normalized separation vector and g_1, q_1 and q_2 are three functions defining the intensity of the dipolar and quadrupolar contributions, with:

$$g_1(r_{ij}) = g(r_{ij}) + \frac{12}{35}q(r_{ij})$$
$$q_1(r_{ij}) = \frac{9}{5}q(r_{ij})$$
$$q_2(r_{ij}) = -\frac{2}{5}q(r_{ij})$$

With the functions $g(r_{ij})$ and $q(r_{ij})$ defined and fitted according to the same Bethe-Slater function used to fit the exchange interaction:

$$J\left(r_{ij}\right) = 4a\left(\frac{r_{ij}}{d}\right)^2\left(1-b\left(\frac{r_{ij}}{d}\right)^2\right)e^{-\left(\frac{r_{ij}}{d}\right)^2}\Theta(R_c - r_{ij})$$

where a, b and d are the three constant coefficients defined in the associated "pair_coeff" command.

The coefficients a, b, and d need to be fitted so that the function above matches with the values of the magneto-elastic constant of the materials at stake.

Examples and more explanations about this function and its parameterization are reported in *(Tranchida)*. More examples of parameterization will be provided in future work.

From this DM interaction, each spin i will be submitted to a magnetic torque ω and its associated atom to a force $\mathbf{F}$ (for spin-lattice calculations only).

More details about the derivation of these torques/forces are reported in *(Tranchida)*.

4.259.4 Restrictions

All the *pair/spin* styles are part of the SPIN package. These styles are only enabled if LAMMPS was built with this package, and if the atom_style "spin" was declared. See the *Build package* page for more info.

4.259.5 Related commands

atom_style spin, *pair_coeff*, *pair_eam*,

4.259.6 Default

none

(Tranchida) Tranchida, Plimpton, Thibaudeau and Thompson, Journal of Computational Physics, 372, 406-425, (2018).

4.260 pair_style srp command

4.261 pair_style srp/react command

4.261.1 Syntax

```
pair_style srp cutoff btype dist keyword value ...
pair_style srp/react cutoff btype dist react-id keyword value ...
```

- cutoff = global cutoff for SRP interactions (distance units)
- btype = bond type (numeric, type label, or wildcard) to apply SRP interactions to
- distance = *min* or *mid*
- react-id = id of either fix bond/break or fix bond/create
- zero or more keyword/value pairs may be appended
- keyword = *exclude*

  bptype value = atom type (numeric or type label) for bond particles
  exclude value = yes or no

4.261.2 Examples

```
pair_style hybrid dpd 1.0 1.0 12345 srp 0.8 1 mid exclude yes
pair_coeff 1 1 dpd 60.0 4.5 1.0
pair_coeff 1 2 none
pair_coeff 2 2 srp 100.0 0.8

pair_style hybrid dpd 1.0 1.0 12345 srp 0.8 * min exclude yes
pair_coeff 1 1 dpd 60.0 50 1.0
pair_coeff 1 2 none
pair_coeff 2 2 srp 40.0

fix       create all bond/create  100  1  2  1.0 1 prob  0.2  19852
pair_style hybrid dpd 1.0 1.0 12345 srp/react 0.8 * min create exclude yes
pair_coeff 1 1 dpd 60.0 50 1.0
pair_coeff 1 2 none
pair_coeff 2 2 srp/react 40.0

pair_style hybrid srp 0.8 2 mid
pair_coeff 1 1 none
pair_coeff 1 2 none
pair_coeff 2 2 srp 100.0 0.8

labelmap bond 1 C-C
pair_style hybrid srp 0.8 C-C mid
```

Description

Style *srp* computes a soft segmental repulsive potential (SRP) that acts between pairs of bonds. This potential is useful for preventing bonds from passing through one another when a soft non-bonded potential acts between beads in, for example, DPD polymer chains. An example input script that uses this command is provided in examples/PACKAGES/srp.

Bonds of specified type *btype* interact with one another through a bond-pairwise potential, such that the force on bond i due to bond j is as follows

$$F_{ij}^{\mathrm{SRP}} = C(1 - r/r_c)\hat{r}_{ij} \qquad r < r_c$$

where r and $\hat{r}_{ij}$ are the distance and unit vector between the two bonds. Note that *btype* can be specified as an asterisk "*", which case the interaction is applied to all bond types. The *mid* option computes r and $\hat{r}_{ij}$ from the midpoint distance between bonds. The *min* option computes r and $\hat{r}_{ij}$ from the minimum distance between bonds. The force acting on a bond is mapped onto the two bond atoms according to the lever rule,

$$F_{i1}^{\mathrm{SRP}} = F_{ij}^{\mathrm{SRP}}(L)$$
$$F_{i2}^{\mathrm{SRP}} = F_{ij}^{\mathrm{SRP}}(1 - L)$$

where L is the normalized distance from the atom to the point of closest approach of bond i and j. The *mid* option takes L as 0.5 for each interaction as described in *(Sirk)*.

The following coefficients must be defined via the *pair_coeff* command as in the examples above, or in the data file or restart file read by the *read_data* or *read_restart* commands:

- C (force units)
- r_c (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

Note

Pair style srp considers each bond of type *btype* to be a fictitious "particle" of type *bptype*, where *bptype* is either the largest atom type in the system, or the type set by the *bptype* flag. Any actual existing particles with this atom type will be deleted at the beginning of a run. This means you must specify the number of types in your system accordingly; usually to be one larger than what would normally be the case, e.g. via the *create_box* or by changing the header in your *data file*. The fictitious "bond particles" are inserted at the beginning of the run, and serve as placeholders that define the position of the bonds. This allows neighbor lists to be constructed and pairwise interactions to be computed in almost the same way as is done for actual particles. Because bonds interact only with other bonds, *pair_style hybrid* should be used to turn off interactions between atom type *bptype* and all other types of atoms. An error will be flagged if *pair_style hybrid* is not used.

Note

If using type labels, the type labels must be defined before calling the *pair_coeff* command.

The optional *exclude* keyword determines if forces are computed between first neighbor (directly connected) bonds. For a setting of *no*, first neighbor forces are computed; for *yes* they are not computed. A setting of *no* cannot be used with the *min* option for distance calculation because the minimum distance between directly connected bonds is zero.

Pair style *srp* turns off normalization of thermodynamic properties by particle number, as if the command *thermo_modify norm no* had been issued.

The pairwise energy associated with style *srp* is shifted to be zero at the cutoff distance r_c.

Added in version 3Aug2022.

Pair style *srp/react* interfaces the pair style *srp* with the bond breaking and formation mechanisms provided by fix *bond/break* and fix *bond/create*, respectively. When using this pair style, whenever a bond breaking (or formation) reaction occurs, the corresponding fictitious particle is deleted (or inserted) during the same simulation time step as the reaction. This is useful in the simulation of reactive systems involving large polymeric molecules *(Palkar)* where the segmental repulsive potential is necessary to minimize topological violations, and also needs to be turned on and off according to the progress of the reaction.

4.261.3 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing.

This pair style does not support the *pair_modify* shift option for the energy of the pair interaction. Note that as discussed above, the energy term is already shifted to be 0.0 at the cutoff distance r_c.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes global and per-atom information to *binary restart files*. Pair srp should be used with *pair_style hybrid*, thus the pair_coeff commands need to be specified in the input script when reading a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.261.4 Restrictions

This pair style is part of the MISC package. It is only enabled if LAMMPS was built with that package. See the Making LAMMPS section for more info.

This pair style must be used with *pair_style hybrid*.

This pair style requires the *newton* command to be *on* for non-bonded interactions.

This pair style is not compatible with *rigid body integrators*

4.261.5 Related commands

pair_style hybrid, *pair_coeff*, *pair dpd*

4.261.6 Default

The default keyword value is exclude = yes.

(Sirk) Sirk TW, Sliozberg YR, Brennan JK, Lisal M, Andzelm JW, J Chem Phys, 136 (13) 134903, 2012.

(Palkar) Palkar V, Kuksenok O, J. Phys. Chem. B, 126 (1), 336-346, 2022

4.262 pair_style sw command

Accelerator Variants: *sw/gpu*, *sw/intel*, *sw/kk*, *sw/omp*

4.263 pair_style sw/mod command

Accelerator Variants: *sw/mod/omp*

4.263.1 Syntax

```
pair_style style keyword values
```

- style = *sw* or *sw/mod*
- keyword = *maxdelcs* or *threebody*

  maxdelcs value = delta1 delta2 (optional, sw/mod only)
    delta1 = The minimum thershold for the variation of cosine of three-body angle
    delta2 = The maximum threshold for the variation of cosine of three-body angle
  threebody value = on or off (optional, sw only)
    on (default) = Compute both the three-body and two-body terms of the potential
    off = Compute only the two-body term of the potential

4.263.2 Examples

```
pair_style sw
pair_coeff * * si.sw Si
pair_coeff * * GaN.sw Ga N Ga

pair_style sw/mod maxdelcs 0.25 0.35
pair_coeff * * tmd.sw.mod Mo S S

pair_style hybrid sw threebody on sw threebody off
pair_coeff * * sw 1 mW_xL.sw mW NULL
pair_coeff 1 2 sw 2 mW_xL.sw mW xL
pair_coeff 2 2 sw 2 mW_xL.sw mW xL
```

4.263.3 Description

The *sw* style computes a 3-body *Stillinger-Weber* potential for the energy E of a system of atoms as

$$E = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j \neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk})$$

$$\phi_2(r_{ij}) = A_{ij}\varepsilon_{ij}\left[B_{ij}(\frac{\sigma_{ij}}{r_{ij}})^{p_{ij}} - (\frac{\sigma_{ij}}{r_{ij}})^{q_{ij}}\right] \exp\left(\frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right)$$

$$\phi_3(r_{ij}, r_{ik}, \theta_{ijk}) = \lambda_{ijk}\varepsilon_{ijk}\left[\cos\theta_{ijk} - \cos\theta_{0ijk}\right]^2 \exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right) \exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik} - a_{ik}\sigma_{ik}}\right)$$

where ϕ_2 is a two-body term and ϕ_3 is a three-body term. The summations in the formula are over all neighbors J and K of atom I within a cutoff distance $a`\sigma$.

Added in version 14Dec2021.

The *sw/mod* style is designed for simulations of materials when distinguishing three-body angles are necessary, such as borophene and transition metal dichalcogenides, which cannot be described by the original code for the Stillinger-Weber potential. For instance, there are several types of angles around each Mo atom in *MoS_2*, and some unnecessary angle types should be excluded in the three-body interaction. Such exclusion may be realized by selecting proper angle types directly. The exclusion of unnecessary angles is achieved here by the cut-off function (*f_C(delta)*), which induces only minimum modifications for LAMMPS.

Validation, benchmark tests, and applications of the *sw/mod* style can be found in *(Jiang2)* and *(Jiang3)*.

The *sw/mod* style computes the energy E of a system of atoms, whose potential function is mostly the same as the Stillinger-Weber potential. The only modification is in the three-body term, where the value of $\delta = \cos\theta_{ijk} - \cos\theta_{0ijk}$ used in the original energy and force expression is scaled by a switching factor $f_C(\delta)$:

$$f_C(\delta) = \begin{cases} 1 & : \quad |\delta| < \delta_1 \\ \frac{1}{2} + \frac{1}{2}\cos\left(\pi\frac{|\delta| - \delta_1}{\delta_2 - \delta_1}\right) & : \quad \delta_1 < |\delta| < \delta_2 \\ 0 & : \quad |\delta| > \delta_2 \end{cases}$$

This cut-off function decreases smoothly from 1 to 0 over the range $[\delta_1, \delta_2]$. This smoothly turns off the energy and force contributions for $|\delta| > \delta_2$. It is suggested that $\delta 1$ and δ_2 to be the value around $0.5\,|\cos\theta_1 - \cos\theta_2|$, with θ_1 and θ_2 as the different types of angles around an atom. For borophene and transition metal dichalcogenides, $\delta_1 = 0.25$ and $\delta_2 = 0.35$. This value enables the cut-off function to exclude unnecessary angles in the three-body SW terms.

Note

The cut-off function is just to be used as a technique to exclude some unnecessary angles, and it has no physical meaning. It should be noted that the force and potential are inconsistent with each other in the decaying range of the cut-off function, as the angle dependence for the cut-off function is not implemented in the force (first derivation of potential). However, the angle variation is much smaller than the given threshold value for actual simulations, so the inconsistency between potential and force can be neglected in actual simulations.

Added in version 3Aug2022.

The *threebody* keyword is optional and determines whether or not the three-body term of the potential is calculated. The default value is "on" and it is only available for the plain *sw* pair style variants, but not available for the *sw/mod* and *sw/angle/table* pair style variants. To turn off the threebody contributions all λ_{ijk} parameters from the potential file are forcibly set to 0. In addition the pair style implementation may employ code optimizations for the *threebody off* setting that can result in significant speedups versus the default. These code optimizations are currently only available for the MANYBODY and OPENMP packages.

Only a single pair_coeff command is used with the *sw* and *sw/mod* styles which specifies a Stillinger-Weber potential file with parameters for all needed elements, except for when the *threebody off* setting is used (see note below). These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of SW elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.sw has Stillinger-Weber values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_style sw
pair_coeff * * SiC.sw Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1, 2, and 3 to the Si element in the SW file. The final C argument maps LAMMPS atom type 4 to the C element in the SW file. If an argument value is specified as NULL, the mapping is not performed. This can be used when an *sw* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Note

When the *threebody off* keyword is used, multiple pair_coeff commands may be used to specific the pairs of atoms which don't require three-body term. In these cases, the first 2 arguments are not required to be * *, the potential parameter file is only read by the first *pair_coeff command* and the element to atom type mappings must be consistent across all *pair_coeff* statements. If not LAMMPS will abort with an error.

Stillinger-Weber files in the *potentials* directory of the LAMMPS distribution have a ".sw" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to the two-body and three-body coefficients in the formula above:

- element 1 (the center atom in a 3-body interaction)
- element 2
- element 3

- ε (energy units)
- σ (distance units)
- a
- λ
- γ
- $\cos \theta_0$
- A
- B
- p
- q
- tol

The A, B, p, and q parameters are used only for two-body interactions. The λ and $\cos \theta_0$ parameters are used only for three-body interactions. The ε, σ and a parameters are used for both two-body and three-body interactions. γ is used only in the three-body interactions, but is defined for pairs of atoms. The non-annotated parameters are unitless.

LAMMPS introduces an additional performance-optimization parameter tol that is used for both two-body and three-body interactions. In the Stillinger-Weber potential, the interaction energies become negligibly small at atomic separations substantially less than the theoretical cutoff distances. LAMMPS therefore defines a virtual cutoff distance based on a user defined tolerance tol. The use of the virtual cutoff distance in constructing atom neighbor lists can significantly reduce the neighbor list sizes and therefore the computational cost. LAMMPS provides a *tol* value for each of the three-body entries so that they can be separately controlled. If tol = 0.0, then the standard Stillinger-Weber cutoff is used.

The Stillinger-Weber potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify SW parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

As annotated above, the first element in the entry is the center atom in a three-body interaction. Thus an entry for SiCC means a Si atom with 2 C atoms as neighbors. The parameter values used for the two-body interaction come from the entry where the second and third elements are the same. Thus the two-body parameters for Si interacting with C, comes from the SiCC entry. The three-body parameters can in principle be specific to the three elements of the configuration. In the literature, however, the three-body parameters are usually defined by simple formulas involving two sets of pairwise parameters, corresponding to the ij and ik pairs, where i is the center atom. The user must ensure that the correct combining rule is used to calculate the values of the three-body parameters for alloys. Note also that the function ϕ_3 contains two exponential screening factors with parameter values from the ij pair and ik pairs. So ϕ_3 for a C atom bonded to a Si atom and a second C atom will depend on the three-body parameters for the CSiC entry, and also on the two-body parameters for the CCC and CSiSi entries. Since the order of the two neighbors is arbitrary, the three-body parameters for entries CSiC and CCSi should be the same. Similarly, the two-body parameters for entries SiCC and CSiSi should also be the same. The parameters used only for two-body interactions (A, B, p, and q) in entries whose second and third element are different (e.g. SiCSi) are not used for anything and can be set to 0.0 if desired. This is also true for the parameters in ϕ_3 that are taken from the ij and ik pairs (σ, a, γ)

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

Note

When using the INTEL package with this style, there is an additional 5 to 10 percent performance improvement when the Stillinger-Weber parameters p and q are set to 4 and 0 respectively. These parameters are common for modeling silicon and water.

4.263.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

The single() function of the *sw* pair style is only enabled and supported for the case of the *threebody off* setting.

4.263.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The Stillinger-Weber potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the sw or sw/mod pair styles with any LAMMPS units, but you would need to create your own SW potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units. If the potential file contains a 'UNITS:' metadata tag in the first line of the potential file, then LAMMPS can convert it transparently between "metal" and "real" units.

4.263.6 Related commands

pair_coeff

4.263.7 Default

The default value for the *threebody* setting of the "sw" pair style is "on", the default values for the "*maxdelcs* setting of the *sw/mod* pair style are *delta1* = 0.25 and *delta2* = 0.35`.

(Stillinger) Stillinger and Weber, Phys Rev B, 31, 5262 (1985).

(Jiang2) J.-W. Jiang, Nanotechnology 26, 315706 (2015).

(Jiang3) J.-W. Jiang, Acta Mech. Solida. Sin 32, 17 (2019).

4.264 pair_style sw/angle/table command

4.264.1 Syntax

```
pair_style style
```

- style = *sw/angle/table*

4.264.2 Examples

```
pair_style sw/angle/table
pair_coeff * * spce.sw type
```

Used in example input script:

```
examples/PACKAGES/manybody_table/in.spce_sw
```

4.264.3 Description

Added in version 2Jun2022.

The *sw/angle/table* style is a modification of the original *pair_style sw*. It has been developed for coarse-grained simulations (of water) (*Scherer1*), but can be employed for all kinds of systems. It computes a modified 3-body *Stillinger-Weber* potential for the energy E of a system of atoms as

$$E = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j \neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk})$$

$$\phi_2(r_{ij}) = A_{ij}\varepsilon_{ij}\left[B_{ij}(\frac{\sigma_{ij}}{r_{ij}})^{p_{ij}} - (\frac{\sigma_{ij}}{r_{ij}})^{q_{ij}}\right] \exp\left(\frac{\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right)$$

$$\phi_3(r_{ij}, r_{ik}, \theta_{ijk}) = f^{3b}\left(\theta_{ijk}\right) \exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij} - a_{ij}\sigma_{ij}}\right) \exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik} - a_{ik}\sigma_{ik}}\right)$$

where ϕ_2 is a two-body term and ϕ_3 is a three-body term. The summations in the formula are over all neighbors J and K of atom I within a cutoff distance $a\sigma$. In contrast to the original *sw* style, *sw/angle/table* allows for a flexible three-body

term $f^{3b}\left(\theta_{ijk}\right)$ which is read in as a tabulated interaction. It can be parameterized with the csg_fmatch app of VOTCA as available at: https://gitlab.mpcdf.mpg.de/votca/votca.

Only a single pair_coeff command is used with the *sw/angle/table* style which specifies a modified Stillinger-Weber potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N_el additional arguments after the ".sw" filename in the pair_coeff command, where N_el is the number of LAMMPS atom types:

- ".sw" filename
- N_el element names = mapping of SW elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.sw has Stillinger-Weber values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.sw Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the SW file. The final C argument maps LAMMPS atom type 4 to the C element in the SW file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *sw/angle/table* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The (modified) Stillinger-Weber files have a ".sw" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to the two-body and three-body coefficients in the formula above. Here, also the suffix ".sw" is used though the original Stillinger-Weber file format is supplemented with four additional lines per parameter block to specify the tabulated three-body interaction. A single entry then contains:

- element 1 (the center atom in a 3-body interaction)
- element 2
- element 3
- ε (energy units)
- σ (distance units)
- a
- λ
- γ
- cos θ_0
- A
- B
- p
- q
- tol
- filename
- keyword
- style
- N

The A, B, p, and q parameters are used only for two-body interactions. The λ and $\cos\theta_0$ parameters, only used for three-body interactions in the original Stillinger-Weber style, are read in but ignored in this modified pair style. The ε parameter is only used for two-body interactions in this modified pair style and not for the three-body terms. The σ and a parameters are used for both two-body and three-body interactions. γ is used only in the three-body interactions, but is defined for pairs of atoms. The non-annotated parameters are unitless.

LAMMPS introduces an additional performance-optimization parameter tol that is used for both two-body and three-body interactions. In the Stillinger-Weber potential, the interaction energies become negligibly small at atomic separations substantially less than the theoretical cutoff distances. LAMMPS therefore defines a virtual cutoff distance based on a user defined tolerance tol. The use of the virtual cutoff distance in constructing atom neighbor lists can significantly reduce the neighbor list sizes and therefore the computational cost. LAMMPS provides a *tol* value for each of the three-body entries so that they can be separately controlled. If tol = 0.0, then the standard Stillinger-Weber cutoff is used.

The additional parameters *filename*, *keyword*, *style*, and *N* refer to the tabulated angular potential $f^{3b}\left(\theta_{ijk}\right)$. The tabulated angular potential has to be of the format as used in the *angle_style table* command:

An interpolation tables of length *N* is created. The interpolation is done in one of 2 *styles*: *linear* or *spline*. For the *linear* style, the angle is used to find 2 surrounding table values from which an energy or its derivative is computed by linear interpolation. For the *spline* style, a cubic spline coefficients are computed and stored at each of the *N* values in the table. The angle is used to find the appropriate set of coefficients which are used to evaluate a cubic polynomial which computes the energy or derivative.

The *filename* specifies the file containing the tabulated energy and derivative values of $f^{3b}\left(\theta_{ijk}\right)$. The *keyword* then specifies a section of the file. The format of this file is as follows (without the parenthesized comments):

```
# Angle potential for harmonic (one or more comment or blank lines)

HAM                         (keyword is the first text on line)
N 181 FP 0 0 EQ 90.0          (N, FP, EQ parameters)
                          (blank line)
1 0.0 200.5 2.5               (index, angle, energy, derivative)
2 1.0 198.0 2.5
...
181 180.0 0.0 0.0
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required and its value is the number of table entries that follow. Note that this may be different than the *N* specified in the Stillinger-Weber potential file. Let Nsw = *N* in the ".sw" file, and Nfile = "N" in the tabulated angular file. What LAMMPS does is a preliminary interpolation by creating splines using the Nfile tabulated values as nodal points. It uses these to interpolate as needed to generate energy and derivative values at Ntable different points. The resulting tables of length Nsw are then used as described above, when computing energy and force for individual angles and their atoms. This means that if you want the interpolation tables of length Nsw to match exactly what is in the tabulated file (with effectively no preliminary interpolation), you should set Nsw = Nfile.

The "FP" parameter is optional. If used, it is followed by two values fplo and fphi, which are the second derivatives at the innermost and outermost angle settings. These values are needed by the spline construction routines. If not specified by the "FP" parameter, they are estimated (less accurately) by the first two and last two derivative values in the table.

The "EQ" parameter is also optional. If used, it is followed by a the equilibrium angle value, which is used, for example, by the *fix shake* command. If not used, the equilibrium angle is set to 180.0.

Following a blank line, the next N lines of the angular table file list the tabulated values. On each line, the first value is

the index from 1 to N, the second value is the angle value (in degrees), the third value is the energy (in energy units), and the fourth is -dE/d(theta) (also in energy units). The third term is the energy of the 3-atom configuration for the specified angle. The last term is the derivative of the energy with respect to the angle (in degrees, not radians). Thus the units of the last term are still energy, not force. The angle values must increase from one line to the next. The angle values must also begin with 0.0 and end with 180.0, i.e. span the full range of possible angles.

Note that one angular potential file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified *keyword* of appropriate section of the ".sw" file.

The Stillinger-Weber potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify SW parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

As annotated above, the first element in the entry is the center atom in a three-body interaction. Thus an entry for SiCC means a Si atom with 2 C atoms as neighbors. The parameter values used for the two-body interaction come from the entry where the second and third elements are the same. Thus the two-body parameters for Si interacting with C, comes from the SiCC entry. The three-body angular potential $f^{3b}\left(\theta_{ijk}\right)$ can in principle be specific to the three elements of the configuration. However, the user must ensure that it makes physically sense. Note also that the function ϕ_3 contains two exponential screening factors with parameter values from the ij pair and ik pairs. So ϕ_3 for a C atom bonded to a Si atom and a second C atom will depend on the three-body parameters for the CSiC entry, and also on the two-body parameters for the CCC and CSiSi entries. Since the order of the two neighbors is arbitrary, the three-body parameters and the tabulated angular potential for entries CSiC and CCSi should be the same. Similarly, the two-body parameters for entries SiCC and CSiSi should also be the same. The parameters used only for two-body interactions (A, B, p, and q) in entries whose second and third element are different (e.g. SiCSi) are not used for anything and can be set to 0.0 if desired. This is also true for the parameters in ϕ_3 that are taken from the ij and ik pairs (σ, a, γ)

Additional input files and reference data can be found at: https://gitlab.mpcdf.mpg.de/votca/votca/-/tree/master/csg-tutorials/spce/3body_sw

---

4.264.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file, but not for the tabulated angular potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.264.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

4.264.6 Related commands

pair_coeff, *pair_style sw*, *pair_style threebody/table*

(Stillinger) Stillinger and Weber, Phys Rev B, 31, 5262 (1985).

(Scherer1) C. Scherer and D. Andrienko, Phys. Chem. Chem. Phys. 20, 22387-22394 (2018).

4.265 pair_style table command

Accelerator Variants: *table/gpu*, *table/kk*, *table/omp*

4.265.1 Syntax

```
pair_style table style N keyword ...
```

- style = *lookup* or *linear* or *spline* or *bitmap* = method of interpolation
- N = use N values in *lookup*, *linear*, *spline* tables
- N = use 2^N values in *bitmap* tables
- zero or more keywords may be appended
- keyword = *ewald* or *pppm* or *msm* or *dispersion* or *tip4p*

4.265.2 Examples

```
pair_style table linear 1000
pair_style table linear 1000 pppm
pair_style table bitmap 12
pair_coeff * 3 morse.table ENTRY1
pair_coeff * 3 morse.table ENTRY1 7.0
```

4.265.3 Description

Style *table* creates interpolation tables from potential energy and force values listed in a file(s) as a function of distance. When performing dynamics or minimization, the interpolation tables are used to evaluate energy and forces for pairwise interactions between particles, similar to how analytic formulas are used for other pair styles.

The interpolation tables are created as a pre-computation by fitting cubic splines to the file values and interpolating energy and force values at each of *N* distances. During a simulation, the tables are used to interpolate energy and force values as needed for each pair of particles separated by a distance *R*. The interpolation is done in one of 4 styles: *lookup*, *linear*, *spline*, or *bitmap*.

For the *lookup* style, the distance *R* is used to find the nearest table entry, which is the energy or force.

For the *linear* style, the distance *R* is used to find the 2 surrounding table values from which an energy or force is computed by linear interpolation.

For the *spline* style, cubic spline coefficients are computed and stored for each of the *N* values in the table, one set of splines for energy, another for force. Note that these splines are different than the ones used to pre-compute the *N* values. Those splines were fit to the *Nfile* values in the tabulated file, where often *Nfile* < *N*. The distance *R* is used to find the appropriate set of spline coefficients which are used to evaluate a cubic polynomial which computes the energy or force.

For the *bitmap* style, the specified *N* is used to create interpolation tables that are 2^N in length. The distance *R* is used to index into the table via a fast bit-mapping technique due to *(Wolff)*, and a linear interpolation is performed between adjacent table values.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- filename
- keyword
- cutoff (distance units)

The filename specifies a file containing tabulated energy and force values. The keyword specifies a section of the file. The cutoff is an optional coefficient. If not specified, the outer cutoff in the table itself (see below) will be used to build an interpolation table that extend to the largest tabulated distance. If specified, only file values up to the cutoff are used to create the interpolation table. The format of this file is described below.

If your tabulated potential(s) are designed to be used as the short-range part of one of the long-range solvers specified by the *kspace_style* command, then you must use one or more of the optional keywords listed above for the pair_style command. These are *ewald* or *pppm* or *msm* or *dispersion* or *tip4p*. This is so LAMMPS can ensure the short-range potential and long-range solver are compatible with each other, as it does for other short-range pair styles, such as *pair_style lj/cut/coul/long*. Note that it is up to you to ensure the tabulated values for each pair of atom types has the correct functional form to be compatible with the matching long-range solver.

Here are some guidelines for using the pair_style table command to best effect:

- Vary the number of table points; you may need to use more than you think to get good resolution.
- Always use the *pair_write* command to produce a plot of what the final interpolated potential looks like. This can show up interpolation "features" you may not like.
- Start with the linear style; it's the style least likely to have problems.
- Use *N* in the pair_style command equal to the "N" in the tabulation file, and use the "RSQ" or "BITMAP" parameter, so additional interpolation is not needed. See discussion below.
- Make sure that your tabulated forces and tabulated energies are consistent (dE/dr = -F) over the entire range of r values. LAMMPS will warn if this is not the case.

- Use as large an inner cutoff as possible. This avoids fitting splines to very steep parts of the potential.

Suitable tables in the correct format for use with these pair styles can be created by LAMMPS itself using the *pair_write* command. In combination with the pair styles *python*, *lepton, or lepton/coul* this can be a powerful mechanism to implement and test tables for use with LAMMPS. Another option to generate tables is the Python code in the tools/tabulate folder of the LAMMPS source code distribution.

The format of a tabulated file has an (optional) header followed by a series of one or more sections, defined as follows (without the parenthesized comments). The header must start with a # character and the DATE: and UNITS: tags will be parsed and used:

```
# DATE: 2020-06-10  UNITS: real  CONTRIBUTOR: ... (header line)
# Morse potential for Fe   (one or more comment or blank lines)

MORSE_FE                    (keyword is first text on line)
N 500 R 1.0 10.0          (N, R, RSQ, BITMAP, FPRIME parameters)
                        (blank)
1 1.0 25.5 102.34         (index, r, energy, force)
2 1.02 23.4 98.5
...
500 10.0 0.001 0.003
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The line can contain additional text, but the initial text must match the argument specified in the pair_coeff command. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required and its value is the number of table entries that follow. Note that this may be different than the *N* specified in the *pair_style table* command. Let Ntable = *N* in the pair_style command, and Nfile = "N" in the tabulated file. What LAMMPS does is a preliminary interpolation by creating splines using the Nfile tabulated values as nodal points. It uses these to interpolate energy and force values at Ntable different points. The resulting tables of length Ntable are then used as described above, when computing energy and force for individual pair distances. This means that if you want the interpolation tables of length Ntable to match exactly what is in the tabulated file (with effectively no preliminary interpolation), you should set Ntable = Nfile, and use the "RSQ" or "BITMAP" parameter. This is because the internal table abscissa is always RSQ (separation distance squared), for efficient lookup.

All other parameters are optional. If "R" or "RSQ" or "BITMAP" does not appear, then the distances in each line of the table are used as-is to perform spline interpolation. In this case, the table values can be spaced in *r* uniformly or however you wish to position table values in regions of large gradients.

If used, the parameters "R" or "RSQ" are followed by 2 values *rlo* and *rhi*. If specified, the distance associated with each energy and force value is computed from these 2 values (at high accuracy), rather than using the (low-accuracy) value listed in each line of the table. The distance values in the table file are ignored in this case. For "R", distances uniformly spaced between *rlo* and *rhi* are computed; for "RSQ", squared distances uniformly spaced between *rlo*rlo* and *rhi*rhi* are computed.

Note

If you use "R" or "RSQ", the tabulated distance values in the file are effectively ignored, and replaced by new values as described in the previous paragraph. If the distance value in the table is not very close to the new value (i.e. round-off difference), then you will be assigning energy/force values to a different distance, which is probably not what you want. LAMMPS will warn if this is occurring.

If used, the parameter "BITMAP" is also followed by 2 values *rlo* and *rhi*. These values, along with the "N" value determine the ordering of the N lines that follow and what distance is associated with each. This ordering is complex, so it is not documented here, since this file is typically produced by the *pair_write* command with its *bitmap* option. When the table is in BITMAP format, the "N" parameter in the file must be equal to 2^M where M is the value specified in the pair_style command. Also, a cutoff parameter cannot be used as an optional third argument in the pair_coeff command; the entire table extent as specified in the file must be used.

If used, the parameter "FPRIME" is followed by 2 values *fplo* and *fphi* which are the derivative of the force at the innermost and outermost distances listed in the table. These values are needed by the spline construction routines. If not specified by the "FPRIME" parameter, they are estimated (less accurately) by the first 2 and last 2 force values in the table. This parameter is not used by BITMAP tables.

Following a blank line, the next N lines list the tabulated values. On each line, the first value is the index from 1 to N, the second value is r (in distance units), the third value is the energy (in energy units), and the fourth is the force (in force units). The r values must increase from one line to the next (unless the BITMAP parameter is specified).

Note that one file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified keyword.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.265.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The *pair_modify* shift, table, and tail options are not relevant for this pair style.

This pair style writes the settings for the "pair_style table" command to *binary restart files*, so a pair_style command does not need to specified in an input script that reads a restart file. However, the coefficient information is not stored in the restart file, since it is tabulated in the potential files. Thus, pair_coeff commands do need to be specified in the restart input script.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.265.5 Restrictions

none

4.265.6 Related commands

pair_coeff, *pair_write*

4.265.7 Default

none

(Wolff) Wolff and Rudd, Comp Phys Comm, 120, 200-32 (1999).

4.266 pair_style table/rx command

Accelerator Variants: *table/rx/kk*

4.266.1 Syntax

```
pair_style table style N ...
```

- style = *lookup* or *linear* or *spline* or *bitmap* = method of interpolation
- N = use N values in *lookup*, *linear*, *spline* tables
- weighting = fractional or molecular (optional)

4.266.2 Examples

```
pair_style table/rx linear 1000
pair_style table/rx linear 1000 fractional
pair_style table/rx linear 1000 molecular
pair_coeff * * rxn.table ENTRY1 h2o h2o 10.0
pair_coeff * * rxn.table ENTRY1 1fluid 1fluid 10.0
pair_coeff * 3 rxn.table ENTRY1 h2o no2 10.0
```

4.266.3 Description

Style *table/rx* is used in reaction DPD simulations,where the coarse-grained (CG) particles are composed of *m* species whose reaction rate kinetics are determined from a set of *n* reaction rate equations through the *fix rx* command. The species of one CG particle can interact with a species in a neighboring CG particle through a site-site interaction potential model. Style *table/rx* creates interpolation tables of length *N* from pair potential and force values listed in a file(s) as a function of distance. The files are read by the *pair_coeff* command.

The interpolation tables are created by fitting cubic splines to the file values and interpolating energy and force values at each of *N* distances. During a simulation, these tables are used to interpolate energy and force values as needed. The interpolation is done in one of 4 styles: *lookup*, *linear*, *spline*, or *bitmap*.

For the *lookup* style, the distance between two atoms is used to find the nearest table entry, which is the energy or force.

For the *linear* style, the pair distance is used to find 2 surrounding table values from which an energy or force is computed by linear interpolation.

For the *spline* style, a cubic spline coefficients are computed and stored at each of the *N* values in the table. The pair distance is used to find the appropriate set of coefficients which are used to evaluate a cubic polynomial which computes the energy or force.

For the *bitmap* style, the N means to create interpolation tables that are 2^N in length. The pair distance is used to index into the table via a fast bit-mapping technique *(Wolff)* and a linear interpolation is performed between adjacent table values.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above.

- filename
- keyword
- species1
- species2
- cutoff (distance units)

The filename specifies a file containing tabulated energy and force values. The keyword specifies a section of the file. The cutoff is an optional coefficient. If not specified, the outer cutoff in the table itself (see below) will be used to build an interpolation table that extend to the largest tabulated distance. If specified, only file values up to the cutoff are used to create the interpolation table. The format of this file is described below.

The species tags define the site-site interaction potential between two species contained within two different particles. The species tags must either correspond to the species defined in the reaction kinetics files specified with the *fix rx* command or they must correspond to the tag "1fluid", signifying interaction with a product species mixture determined through a one-fluid approximation. The interaction potential is weighted by the geometric average of either the mole fraction concentrations or the number of molecules associated with the interacting coarse-grained particles (see the *fractional* or *molecular* weighting pair style options). The coarse-grained potential is stored before and after the reaction kinetics solver is applied, where the difference is defined to be the internal chemical energy (uChem).

Here are some guidelines for using the pair_style table/rx command to best effect:

- Vary the number of table points; you may need to use more than you think to get good resolution.
- Always use the *pair_write* command to produce a plot of what the final interpolated potential looks like. This can show up interpolation "features" you may not like.
- Start with the linear style; it's the style least likely to have problems.
- Use *N* in the pair_style command equal to the "N" in the tabulation file, and use the "RSQ" or "BITMAP" parameter, so additional interpolation is not needed. See discussion below.
- Make sure that your tabulated forces and tabulated energies are consistent (dE/dr = -F) along the entire range of r values.
- Use as large an inner cutoff as possible. This avoids fitting splines to very steep parts of the potential.

The format of a tabulated file is a series of one or more sections, defined as follows (without the parenthesized comments):

```
# Morse potential for Fe   (one or more comment or blank lines)

MORSE_FE                   (keyword is first text on line)
N 500 R 1.0 10.0          (N, R, RSQ, BITMAP, FPRIME parameters)
                        (blank)
1 1.0 25.5 102.34         (index, r, energy, force)
2 1.02 23.4 98.5
...
500 10.0 0.001 0.003
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The line can contain additional text, but the initial text must match the argument specified in the pair_coeff command. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required and its value is the number of table entries that follow. Note that this may be different than the *N* specified in the *pair_style table/rx* command. Let Ntable = *N* in the pair_style command, and Nfile = "N" in the tabulated file. What LAMMPS does is a preliminary interpolation by creating splines using the Nfile tabulated values as nodal points. It uses these to interpolate as needed to generate energy and force values at Ntable different points. The resulting tables of length Ntable are then used as described above, when computing energy and force for individual pair distances. This means that if you want the interpolation tables of length Ntable to match exactly what is in the tabulated file (with effectively no preliminary interpolation), you should set Ntable = Nfile, and use the "RSQ" or "BITMAP" parameter. The internal table abscissa is RSQ (separation distance squared).

All other parameters are optional. If "R" or "RSQ" or "BITMAP" does not appear, then the distances in each line of the table are used as-is to perform spline interpolation. In this case, the table values can be spaced in *r* uniformly or however you wish to position table values in regions of large gradients.

If used, the parameters "R" or "RSQ" are followed by 2 values *rlo* and *rhi*. If specified, the distance associated with each energy and force value is computed from these 2 values (at high accuracy), rather than using the (low-accuracy) value listed in each line of the table. The distance values in the table file are ignored in this case. For "R", distances uniformly spaced between *rlo* and *rhi* are computed; for "RSQ", squared distances uniformly spaced between *rlo***rlo* and *rhi***rhi* are computed.

If used, the parameter "BITMAP" is also followed by 2 values *rlo* and *rhi*. These values, along with the "N" value determine the ordering of the N lines that follow and what distance is associated with each. This ordering is complex, so it is not documented here, since this file is typically produced by the *pair_write* command with its *bitmap* option. When the table is in BITMAP format, the "N" parameter in the file must be equal to 2^M where M is the value specified in the pair_style command. Also, a cutoff parameter cannot be used as an optional third argument in the pair_coeff command; the entire table extent as specified in the file must be used.

If used, the parameter "FPRIME" is followed by 2 values *fplo* and *fphi* which are the derivative of the force at the innermost and outermost distances listed in the table. These values are needed by the spline construction routines. If not specified by the "FPRIME" parameter, they are estimated (less accurately) by the first 2 and last 2 force values in the table. This parameter is not used by BITMAP tables.

Following a blank line, the next N lines list the tabulated values. On each line, the first value is the index from 1 to N, the second value is r (in distance units), the third value is the energy (in energy units), and the fourth is the force (in force units). The r values must increase from one line to the next (unless the BITMAP parameter is specified).

Note that one file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified keyword.

4.266.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The *pair_modify* shift, table, and tail options are not relevant for this pair style.

This pair style writes the settings for the "pair_style table/rx" command to *binary restart files*, so a pair_style command does not need to specified in an input script that reads a restart file. However, the coefficient information is not stored in the restart file, since it is tabulated in the potential files. Thus, pair_coeff commands do need to be specified in the restart input script.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.266.5 Restrictions

This command is part of the DPD-REACT package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.266.6 Related commands

pair_coeff

4.266.7 Default

fractional weighting

(Wolff) Wolff and Rudd, Comp Phys Comm, 120, 200-32 (1999).

4.267 pair_style tersoff command

Accelerator Variants: *tersoff/gpu*, *tersoff/intel*, *tersoff/kk*, *tersoff/omp*

4.268 pair_style tersoff/table command

Accelerator Variants: *tersoff/table/omp*

4.268.1 Syntax

```
pair_style style keywords values
```

- style = *tersoff* or *tersoff/table*
- keyword = *shift*

  shift value = delta
    delta = negative shift in equilibrium bond length

4.268.2 Examples

```
pair_style tersoff
pair_coeff * * Si.tersoff Si
pair_coeff * * SiC.tersoff Si C Si

pair_style tersoff/table
pair_coeff * * SiCGe.tersoff Si(D)

pair_style tersoff shift 0.05
pair_coeff * * Si.tersoff Si
```

4.268.3 Description

The *tersoff* style computes a 3-body Tersoff potential *(Tersoff_1)* for the energy E of a system of atoms as

$$
\begin{aligned}
E &= \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} &= f_C(r_{ij}+\delta)\left[f_R(r_{ij}+\delta) + b_{ij} f_A(r_{ij}+\delta)\right] \\
f_C(r) &= \begin{cases} 1 & : \quad r < R-D \\ \frac{1}{2} - \frac{1}{2}\sin\left(\frac{\pi}{2}\frac{r-R}{D}\right) & : \quad R-D < r < R+D \\ 0 & : \quad r > R+D \end{cases} \\
f_R(r) &= A\exp(-\lambda_1 r) \\
f_A(r) &= -B\exp(-\lambda_2 r) \\
b_{ij} &= \left(1+\beta^n {\zeta_{ij}}^n\right)^{-\frac{1}{2n}} \\
\zeta_{ij} &= \sum_{k \neq i,j} f_C(r_{ik}+\delta) g\left[\theta_{ijk}(r_{ij}, r_{ik})\right] \exp\left[{\lambda_3}^m (r_{ij}-r_{ik})^m\right] \\
g(\theta) &= \gamma_{ijk}\left(1 + \frac{c^2}{d^2} - \frac{c^2}{\left[d^2 + (\cos\theta - \cos\theta_0)^2\right]}\right)
\end{aligned}
$$

where f_R is a two-body term and f_A includes three-body interactions. The summations in the formula are over all neighbors J and K of atom I within a cutoff distance = R + D. δ is an optional negative shift of the equilibrium bond length, as described below.

The *tersoff/table* style uses tabulated forms for the two-body, environment and angular functions. Linear interpolation is performed between adjacent table entries. The table length is chosen to be accurate within 10^-6 with respect to the *tersoff* style energy. The *tersoff/table* should give better performance in terms of speed.

Only a single pair_coeff command is used with the *tersoff* style which specifies a Tersoff potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of Tersoff elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine the SiC.tersoff file has Tersoff values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.tersoff Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the Tersoff file. The final C argument maps LAMMPS atom type 4 to the C element in the Tersoff file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *tersoff* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Tersoff files in the *potentials* directory of the LAMMPS distribution have a ".tersoff" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to coefficients in the formula above:

- element 1 (the center atom in a 3-body interaction)
- element 2 (the atom bonded to the center atom)
- element 3 (the atom influencing the 1-2 bond in a bond-order sense)

- m
- γ
- λ_3 (1/distance units)
- c
- d
- cos θ_0 (can be a value < -1 or > 1)
- n
- β
- λ_2 (1/distance units)
- B (energy units)
- R (distance units)
- D (distance units)
- λ_1 (1/distance units)
- A (energy units)

The n, β, λ_2, B, λ_1, and A parameters are only used for two-body interactions. The m, γ, λ_3, c, d, and cos θ_0 parameters are only used for three-body interactions. The R and D parameters are used for both two-body and three-body interactions. The non-annotated parameters are unitless. The value of m must be 3 or 1.

The Tersoff potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify Tersoff parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

As annotated above, the first element in the entry is the center atom in a three-body interaction and it is bonded to the second atom and the bond is influenced by the third atom. Thus an entry for SiCC means Si bonded to a C with another C atom influencing the bond. Thus three-body parameters for SiCSi and SiSiC entries will not, in general, be the same. The parameters used for the two-body interaction come from the entry where the second element is repeated. Thus the two-body parameters for Si interacting with C, comes from the SiCC entry.

The parameters used for a particular three-body interaction come from the entry with the corresponding three elements. The parameters used only for two-body interactions (n, β, λ_2, B, λ_1, and A) in entries whose second and third element are different (e.g. SiCSi) are not used for anything and can be set to 0.0 if desired.

Note that the twobody parameters in entries such as SiCC and CSiSi are often the same, due to the common use of symmetric mixing rules, but this is not always the case. For example, the beta and n parameters in Tersoff_2 *(Tersoff_2)* are not symmetric. Similarly, the threebody parameters in entries such as SiCSi and SiSiC are often the same, but this is not always the case, particularly the value of R, which is sometimes typed on the first and second elements, sometimes on the first and third elements. Hence the need to specify R and D explicitly for all element triples. For example, while Tersoff's notation in Tersoff_2 *(Tersoff_2)* is ambiguous on this point, and properties of the zincblende lattice are the same for either choice, Tersoff's results for rocksalt are consistent with typing on the first and third elements. *Albe et al.* adopts the same convention. Conversely, the potential for B/N/C from the Cagin group uses the opposite convention, typing on the first and second elements.

We chose the above form so as to enable users to define all commonly used variants of the Tersoff potential. In particular, our form reduces to the original Tersoff form when m = 3 and gamma = 1, while it reduces to the form of *Albe et al.* when beta = 1 and m = 1. Note that in the current Tersoff implementation in LAMMPS, m must be specified as either

3 or 1. Tersoff used a slightly different but equivalent form for alloys, which we will refer to as Tersoff_2 potential *(Tersoff_2)*. The *tersoff/table* style implements Tersoff_2 parameterization only.

LAMMPS parameter values for Tersoff_2 can be obtained as follows: $\gamma_{ijk} = \omega_{ik}$, $\lambda_3 = 0$ and the value of m has no effect. The parameters for species i and j can be calculated using the Tersoff_2 mixing rules:

$$\lambda_1^{i,j} = \frac{1}{2}(\lambda_1^i + \lambda_1^j)$$
$$\lambda_2^{i,j} = \frac{1}{2}(\lambda_2^i + \lambda_2^j)$$
$$A_{i,j} = (A_i A_j)^{1/2}$$
$$B_{i,j} = \chi_{ij}(B_i B_j)^{1/2}$$
$$R_{i,j} = (R_i R_j)^{1/2}$$
$$S_{i,j} = (S_i S_j)^{1/2}$$

Tersoff_2 parameters R and S must be converted to the LAMMPS parameters R and D (R is different in both forms), using the following relations: R=(R'+S')/2 and D=(S'-R')/2, where the primes indicate the Tersoff_2 parameters.

In the potentials directory, the file SiCGe.tersoff provides the LAMMPS parameters for Tersoff's various versions of Si, as well as his alloy parameters for Si, C, and Ge. This file can be used for pure Si, (three different versions), pure C, pure Ge, binary SiC, and binary SiGe. LAMMPS will generate an error if this file is used with any combination involving C and Ge, since there are no entries for the GeC interactions (Tersoff did not publish parameters for this cross-interaction.) Tersoff files are also provided for the SiC alloy (SiC.tersoff) and the GaN (GaN.tersoff) alloys.

Many thanks to Rutuparna Narulkar, David Farrell, and Xiaowang Zhou for helping clarify how Tersoff parameters for alloys have been defined in various papers.

The *shift* keyword computes the energy E of a system of atoms, whose formula is the same as the Tersoff potential. The only modification is that the original equilibrium bond length (r_0) of the system is shifted to $r_0 - \delta$. The minus sign arises because each radial distance r is replaced by $r + \delta$.

The *shift* keyword is designed for simulations of closely matched van der Waals heterostructures. For instance, consider the case of a system with few-layers graphene atop a thick hexagonal boron nitride (h-BN) substrate simulated using periodic boundary conditions. The experimental lattice mismatch of ~1.8% between graphene and h-BN is not well captured by the equilibrium lattice constants of available potentials, thus a small in-plane strain will be introduced in the system when building a periodic supercell. To minimize the effect of strain on simulation results, the *shift* keyword allows adjusting the equilibrium bond length of one of the two materials (e.g., h-BN). Validation, benchmark tests, and applications of the *shift* keyword can be found in *(Mandelli_1)* and *(Ouyang_1)*.

For the specific case discussed above, the force field can be defined as

```
pair_style  hybrid/overlay rebo tersoff shift -0.00407 ilp/graphene/hbn 16.0 coul/shield 16.0
pair_coeff  * * rebo             CH.rebo      NULL NULL C
pair_coeff  * * tersoff          BNC.tersoff B    N    NULL
pair_coeff  * * ilp/graphene/hbn BNCH.ILP    B    N    C
pair_coeff  1 1 coul/shield 0.70
pair_coeff  1 2 coul/shield 0.695
pair_coeff  2 2 coul/shield 0.69
```

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.268.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.268.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The *shift* keyword is not supported by the *tersoff/gpu*, *tersoff/intel*, *tersoff/kk*, *tersoff/table* or *tersoff/table/omp* variants.

The *tersoff/intel* pair style is only available when compiling LAMMPS with the Intel compilers.

The Tersoff potential files provided with LAMMPS (see the potentials directory) are parameterized for *"metal" units*. In addition the pair style supports converting potential parameters on-the-fly between "metal" and "real" units. You can use the *tersoff* pair style variants with any LAMMPS units setting, but you would need to create your own Tersoff potential file with coefficients listed in the appropriate units if your simulation does not use "metal" or "real" units.

4.268.6 Related commands

pair_coeff

4.268.7 Default

shift delta = 0.0

(Tersoff_1) J. Tersoff, Phys Rev B, 37, 6991 (1988).

(Albe) J. Nord, K. Albe, P. Erhart, and K. Nordlund, J. Phys.: Condens. Matter, 15, 5649(2003).

(Tersoff_2) J. Tersoff, Phys Rev B, 39, 5566 (1989); errata (PRB 41, 3248)

(Mandelli_1) D. Mandelli, W. Ouyang, M. Urbakh, and O. Hod, ACS Nano 13(7), 7603-7609 (2019).

(Ouyang_1) W. Ouyang et al., J. Chem. Theory Comput. 16(1), 666-676 (2020).

4.269 pair_style tersoff/mod command

Accelerator Variants: *tersoff/mod/gpu*, *tersoff/mod/kk*, *tersoff/mod/omp*

4.270 pair_style tersoff/mod/c command

Accelerator Variants: *tersoff/mod/c/omp*

4.270.1 Syntax

```
pair_style style keywords values
```

- style = *tersoff/mod* or *tersoff/mod/c*
- keyword = *shift*

  shift value = delta
    delta = negative shift in equilibrium bond length

4.270.2 Examples

```
pair_style tersoff/mod
pair_coeff * * Si.tersoff.mod Si Si

pair_style tersoff/mod/c
pair_coeff * * Si.tersoff.modc Si Si
```

4.270.3 Description

The *tersoff/mod* and *tersoff/mod/c* styles computes a bond-order type interatomic potential *(Kumagai)* based on a 3-body Tersoff potential *(Tersoff_1)*, *(Tersoff_2)* with modified cutoff function and angular-dependent term, giving the energy E of a system of atoms as

$$
\begin{aligned}
E &= \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} &= f_C(r_{ij}+\delta)\left[f_R(r_{ij}+\delta) + b_{ij} f_A(r_{ij}+\delta)\right] \\
f_C(r) &= \begin{cases} 1 & : \quad r < R-D \\ \frac{1}{2} - \frac{9}{16}\sin\left(\frac{\pi}{2}\frac{r-R}{D}\right) - \frac{1}{16}\sin\left(\frac{3\pi}{2}\frac{r-R}{D}\right) & : \quad R-D < r < R+D \\ 0 & : \quad r > R+D \end{cases} \\
f_R(r) &= A\exp(-\lambda_1 r) \\
f_A(r) &= -B\exp(-\lambda_2 r) \\
b_{ij} &= \left(1+{\zeta_{ij}}^{\eta}\right)^{-\frac{1}{2n}} \\
\zeta_{ij} &= \sum_{k \neq i,j} f_C(r_{ik}+\delta) g(\theta_{ijk}) \exp\left[\alpha(r_{ij}-r_{ik})^{\beta}\right] \\
g(\theta) &= c_1 + g_o(\theta) g_a(\theta) \\
g_o(\theta) &= \frac{c_2(h-\cos\theta)^2}{c_3 + (h-\cos\theta)^2} \\
g_a(\theta) &= 1 + c_4 \exp\left[-c_5(h-\cos\theta)^2\right]
\end{aligned}
$$

where f_R is a two-body term and f_A includes three-body interactions. δ is an optional negative shift of the equilibrium bond length, as described below.

The summations in the formula are over all neighbors J and K of atom I within a cutoff distance = R + D. The *tersoff/mod/c* style differs from *tersoff/mod* only in the formulation of the V_ij term, where it contains an additional c0 term.

$$
V_{ij} = f_C(r_{ij}+\delta)\left[f_R(r_{ij}+\delta) + b_{ij} f_A(r_{ij}+\delta) + c_0\right]
$$

The modified cutoff function f_C proposed by *(Murty)* and having a continuous second-order differential is employed. The angular-dependent term $g(\theta)$ was modified to increase the flexibility of the potential.

The *tersoff/mod* potential is fitted to both the elastic constants and melting point by employing the modified Tersoff potential function form in which the angular-dependent term is improved. The model performs extremely well in describing the crystalline, liquid, and amorphous phases *(Schelling)*.

Only a single pair_coeff command is used with the *tersoff/mod* style which specifies a Tersoff/MOD potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of Tersoff/MOD elements to atom types

As an example, imagine the Si.tersoff_mod file has Tersoff values for Si. If your LAMMPS simulation has 3 Si atoms types, you would use the following pair_coeff command:

```
pair_coeff * * Si.tersoff_mod Si Si Si
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the Tersoff/MOD file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *tersoff/mod* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Tersoff/MOD file in the *potentials* directory of the LAMMPS distribution have a ".tersoff.mod" suffix. Potential files for the *tersoff/mod/c* style have the suffix ".tersoff.modc". Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to coefficients in the formulae above:

- element 1 (the center atom in a 3-body interaction)
- element 2 (the atom bonded to the center atom)
- element 3 (the atom influencing the 1-2 bond in a bond-order sense)
- β
- α
- h
- η
- β_{ters} = 1 (dummy parameter)
- λ_2 (1/distance units)
- B (energy units)
- R (distance units)
- D (distance units)
- λ_1 (1/distance units)
- A (energy units)
- n
- c1
- c2
- c3
- c4
- c5
- c0 (energy units, tersoff/mod/c only)

The n, η, λ_2, B, λ_1, and A parameters are only used for two-body interactions. The β, α, c1, c2, c3, c4, c5, h parameters are only used for three-body interactions. The R and D parameters are used for both two-body and three-body interactions. The c0 term applies to *tersoff/mod/c* only. The non-annotated parameters are unitless.

The Tersoff/MOD potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). As annotated above, the first element in the entry is the center atom in a three-body interaction and it is bonded to the second atom and the bond is influenced by the third atom. Thus an entry for SiSiSi means Si bonded to a Si with another Si atom influencing the bond.

The *shift* keyword computes the energy E of a system of atoms, whose formula is the same as the Tersoff potential. The only modification is that the original equilibrium bond length (r_0) of the system is shifted to $r_0 - \delta$. The minus sign arises because each radial distance r is replaced by $r + \delta$. More information on this option is given on the main *pair_tersoff* page.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.270.4 Mixing, shift, table, tail correction, restart, rRESPA info

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.270.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The *shift* keyword is not supported by the *tersoff/gpu*, *tersoff/intel*, *tersoff/kk*, *tersoff/table* or *tersoff/table/omp* variants.

The *tersoff/mod* potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the *tersoff/mod* pair style with any LAMMPS units, but you would need to create your own Tersoff/MOD potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.270.6 Related commands

pair_coeff

4.270.7 Default

none

(Kumagai) T. Kumagai, S. Izumi, S. Hara, S. Sakai, Comp. Mat. Science, 39, 457 (2007).

(Tersoff_1) J. Tersoff, Phys Rev B, 37, 6991 (1988).

(Tersoff_2) J. Tersoff, Phys Rev B, 38, 9902 (1988).

(Murty) M.V.R. Murty, H.A. Atwater, Phys Rev B, 51, 4889 (1995).

(Schelling) Patrick K. Schelling, Comp. Mat. Science, 44, 274 (2008).

4.271 pair_style tersoff/zbl command

Accelerator Variants: *tersoff/zbl/gpu*, *tersoff/zbl/kk*, *tersoff/zbl/omp*

4.271.1 Syntax

```
pair_style tersoff/zbl keywords values
```

- keyword = *shift*

  shift value = delta
    delta = negative shift in equilibrium bond length

4.271.2 Examples

```
pair_style tersoff/zbl
pair_coeff * * SiC.tersoff.zbl Si C Si
```

4.271.3 Description

The *tersoff/zbl* style computes a 3-body Tersoff potential *(Tersoff_1)* with a close-separation pairwise modification based on a Coulomb potential and the Ziegler-Biersack-Littmark universal screening function *(ZBL)*, giving the energy

E of a system of atoms as

$$
\begin{aligned}
E &= \frac{1}{2}\sum_i \sum_{j \neq i} V_{ij} \\
V_{ij} &= (1 - f_F(r_{ij} + \delta))V^{ZBL}(r_{ij} + \delta) + f_F(r_{ij} + \delta)V^{Tersoff}(r_{ij} + \delta) \\
f_F(r) &= \frac{1}{1 + e^{-A_F(r - r_C)}}
\end{aligned}
$$

$$
\begin{aligned}
V^{ZBL}(r) &= \frac{1}{4\pi\varepsilon_0}\frac{Z_1 Z_2\, e^2}{r}\phi(r/a) \\
a &= \frac{0.8854\, a_0}{Z_1^{0.23} + Z_2^{0.23}} \\
\phi(x) &= 0.1818e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x}
\end{aligned}
$$

$$
\begin{aligned}
V^{Tersoff}(r) &= f_C(r)\left[f_R(r) + b_{ij} f_A(r)\right] \\
f_C(r) &= \begin{cases} 1 & : \quad r < R - D \\ \frac{1}{2} - \frac{1}{2}\sin\left(\frac{\pi}{2}\frac{r-R}{D}\right) & : \quad R - D < r < R + D \\ 0 & : \quad r > R + D \end{cases} \\
f_R(r) &= A\exp(-\lambda_1 r) \\
f_A(r) &= -B\exp(-\lambda_2 r) \\
b_{ij} &= \left(1 + \beta^n {\zeta_{ij}}^n\right)^{-\frac{1}{2n}} \\
\zeta_{ij} &= \sum_{k \neq i,j} f_C(r_{ik} + \delta) g(\theta_{ijk}) \exp\left[{\lambda_3}^m (r_{ij} - r_{ik})^m\right] \\
g(\theta) &= \gamma_{ijk}\left(1 + \frac{c^2}{d^2} - \frac{c^2}{\left[d^2 + (\cos\theta - \cos\theta_0)^2\right]}\right)
\end{aligned}
$$

The f_F term is a fermi-like function used to smoothly connect the ZBL repulsive potential with the Tersoff potential. There are 2 parameters used to adjust it: A_F and r_C. A_F controls how "sharp" the transition is between the two, and r_C is essentially the cutoff for the ZBL potential.

For the ZBL portion, there are two terms. The first is the Coulomb repulsive term, with Z1, Z2 as the number of protons in each nucleus, e as the electron charge (1 for metal and real units) and ε_0 as the permittivity of vacuum. The second part is the ZBL universal screening function, with a0 being the Bohr radius (typically 0.529 Angstroms), and the remainder of the coefficients provided by the original paper. This screening function should be applicable to most systems. However, it is only accurate for small separations (i.e. less than 1 Angstrom).

For the Tersoff portion, f_R is a two-body term and f_A includes three-body interactions. The summations in the formula are over all neighbors J and K of atom I within a cutoff distance = R + D.

δ is an optional negative shift of the equilibrium bond length, as described below.

Only a single pair_coeff command is used with the *tersoff/zbl* style which specifies a Tersoff/ZBL potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of Tersoff/ZBL elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine the SiC.tersoff.zbl file has Tersoff/ZBL values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.tersoff Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the Tersoff/ZBL file. The final C argument maps LAMMPS atom type 4 to the C element in the Tersoff/ZBL file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *tersoff/zbl* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Tersoff/ZBL files in the *potentials* directory of the LAMMPS distribution have a ".tersoff.zbl" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to coefficients in the formula above:

- element 1 (the center atom in a 3-body interaction)
- element 2 (the atom bonded to the center atom)
- element 3 (the atom influencing the 1-2 bond in a bond-order sense)
- m
- γ
- λ_3 (1/distance units)
- c
- d
- $\cos \theta_0$ (can be a value < -1 or > 1)
- n
- β
- λ_2 (1/distance units)
- B (energy units)
- R (distance units)
- D (distance units)
- λ_1 (1/distance units)
- A (energy units)
- Z_i
- Z_j
- ZBLcut (distance units)
- ZBLexpscale (1/distance units)

The n, β, λ_2, B, λ_1, and A parameters are only used for two-body interactions. The m, γ, λ_3, c, d, and $\cos \theta_0$ parameters are only used for three-body interactions. The R and D parameters are used for both two-body and three-body interactions. The Z_i, Z_j, ZBLcut, ZBLexpscale parameters are used in the ZBL repulsive portion of the potential and in the Fermi-like function. The non-annotated parameters are unitless. The value of m must be 3 or 1.

The Tersoff/ZBL potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify Tersoff parameters for

all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

As annotated above, the first element in the entry is the center atom in a three-body interaction and it is bonded to the second atom and the bond is influenced by the third atom. Thus an entry for SiCC means Si bonded to a C with another C atom influencing the bond. Thus three-body parameters for SiCSi and SiSiC entries will not, in general, be the same. The parameters used for the two-body interaction come from the entry where the second element is repeated. Thus the two-body parameters for Si interacting with C, comes from the SiCC entry.

The parameters used for a particular three-body interaction come from the entry with the corresponding three elements. The parameters used only for two-body interactions (n, β, λ_2, B, λ_1, and A) in entries whose second and third element are different (e.g. SiCSi) are not used for anything and can be set to 0.0 if desired.

Note that the twobody parameters in entries such as SiCC and CSiSi are often the same, due to the common use of symmetric mixing rules, but this is not always the case. For example, the beta and n parameters in Tersoff_2 *(Tersoff_2)* are not symmetric.

We chose the above form so as to enable users to define all commonly used variants of the Tersoff portion of the potential. In particular, our form reduces to the original Tersoff form when m = 3 and gamma = 1, while it reduces to the form of *Albe et al.* when beta = 1 and m = 1. Note that in the current Tersoff implementation in LAMMPS, m must be specified as either 3 or 1. Tersoff used a slightly different but equivalent form for alloys, which we will refer to as Tersoff_2 potential *(Tersoff_2)*.

LAMMPS parameter values for Tersoff_2 can be obtained as follows: $\gamma = \omega_{ijk}$, $\lambda_3 = 0$ and the value of m has no effect. The parameters for species i and j can be calculated using the Tersoff_2 mixing rules:

$$\lambda_1^{i,j} = \frac{1}{2}(\lambda_1^i + \lambda_1^j)$$
$$\lambda_2^{i,j} = \frac{1}{2}(\lambda_2^i + \lambda_2^j)$$
$$A_{i,j} = (A_i A_j)^{1/2}$$
$$B_{i,j} = \chi_{ij}(B_i B_j)^{1/2}$$
$$R_{i,j} = (R_i R_j)^{1/2}$$
$$S_{i,j} = (S_i S_j)^{1/2}$$

Tersoff_2 parameters R and S must be converted to the LAMMPS parameters R and D (R is different in both forms), using the following relations: R=(R'+S')/2 and D=(S'-R')/2, where the primes indicate the Tersoff_2 parameters.

In the potentials directory, the file SiCGe.tersoff provides the LAMMPS parameters for Tersoff's various versions of Si, as well as his alloy parameters for Si, C, and Ge. This file can be used for pure Si, (three different versions), pure C, pure Ge, binary SiC, and binary SiGe. LAMMPS will generate an error if this file is used with any combination involving C and Ge, since there are no entries for the GeC interactions (Tersoff did not publish parameters for this cross-interaction.) Tersoff files are also provided for the SiC alloy (SiC.tersoff) and the GaN (GaN.tersoff) alloys.

Many thanks to Rutuparna Narulkar, David Farrell, and Xiaowang Zhou for helping clarify how Tersoff parameters for alloys have been defined in various papers. Also thanks to Ram Devanathan for providing the base ZBL implementation.

The *shift* keyword computes the energy E of a system of atoms, whose formula is the same as the Tersoff potential. The only modification is that the original equilibrium bond length (r_0) of the system is shifted to $r_0 - \delta$. The minus sign arises because each radial distance r is replaced by $r + \delta$. More information on this option is given on the main *pair_tersoff* page.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.271.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.271.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

The *shift* keyword is currently not supported for the *tersoff/gpu* and *tersoff/kk* variants of this pair style.

The tersoff/zbl potential files provided with LAMMPS (see the potentials directory) are parameterized for *"metal" units*. Also the pair style supports converting potential file parameters on-the-fly between "metal" and "real" units. You can use the tersoff/zbl pair style with any LAMMPS units, but you would need to create your own tersoff/zbl potential file with coefficients listed in the appropriate units if your simulation does not use "metal" or "real" units.

4.271.6 Related commands

pair_coeff

4.271.7 Default

none

(Tersoff_1) J. Tersoff, Phys Rev B, 37, 6991 (1988).

(ZBL) J.F. Ziegler, J.P. Biersack, U. Littmark, 'Stopping and Ranges of Ions in Matter' Vol 1, 1985, Pergamon Press.

(Albe) J. Nord, K. Albe, P. Erhart and K. Nordlund, J. Phys.: Condens. Matter, 15, 5649(2003).

(Tersoff_2) J. Tersoff, Phys Rev B, 39, 5566 (1989); errata (PRB 41, 3248)

4.272 pair_style thole command

4.273 pair_style lj/cut/thole/long command

Accelerator Variants: *lj/cut/thole/long/omp*

4.273.1 Syntax

```
pair_style style args
```

- style = *thole* or *lj/cut/thole/long*
- args = list of arguments for a particular style

thole args = damp cutoff
  damp = global damping parameter
  cutoff = global cutoff (distance units)
lj/cut/thole/long args = damp cutoff (cutoff2)
  damp = global damping parameter
  cutoff = global cutoff for LJ (and Thole if only 1 arg) (distance units)
  cutoff2 = global cutoff for Thole (optional) (distance units)

4.273.2 Examples

```
pair_style hybrid/overlay ... thole 2.6 12.0
pair_coeff 1 1 thole 1.0
pair_coeff 1 2 thole 1.0 2.6 10.0
pair_coeff * 2 thole 1.0 2.6

pair_style lj/cut/thole/long 2.6 12.0
```

Example input scripts available: examples/PACKAGES/drude

4.273.3 Description

The *thole* pair styles are meant to be used with force fields that include explicit polarization through Drude dipoles. This link describes how to use the *thermalized Drude oscillator model* in LAMMPS and polarizable models in LAMMPS are discussed on the *Howto polarizable* doc page.

The *thole* pair style should be used as a sub-style within in the *pair_style hybrid/overlay* command, in conjunction with a main pair style including Coulomb interactions, i.e. any pair style containing *coul/cut* or *coul/long* in its style name.

The *lj/cut/thole/long* pair style is equivalent to, but more convenient that the frequent combination *hybrid/overlay lj/cut/coul/long cutoff thole damp cutoff2*. It is not only a shorthand for this pair_style combination, but it also allows for mixing pair coefficients instead of listing them all. The *lj/cut/thole/long* pair style is also a bit faster because it avoids an overlay and can benefit from OMP acceleration. Moreover, it uses a more precise approximation of the direct Coulomb interaction at short range similar to *coul/long/cs*, which stabilizes the temperature of Drude particles.

The *thole* pair styles compute the Coulomb interaction damped at short distances by a function

$$T_{ij}(r_{ij}) = 1 - \left(1 + \frac{s_{ij} r_{ij}}{2}\right) \exp\left(-s_{ij} r_{ij}\right)$$

This function results from an adaptation to point charges *(Noskov)* of the dipole screening scheme originally proposed by *Thole*. The scaling coefficient s_{ij} is determined by the polarizability of the atoms, α_i, and by a Thole damping parameter a. This Thole damping parameter usually takes a value of 2.6, but in certain force fields the value can depend upon the atom types. The mixing rule for Thole damping parameters is the arithmetic average, and for polarizabilities the geometric average between the atom-specific values.

$$s_{ij} = \frac{a_{ij}}{(\alpha_{ij})^{1/3}} = \frac{(a_i + a_j)/2}{[(\alpha_i \alpha_j)^{1/2}]^{1/3}}$$

The damping function is only applied to the interactions between the point charges representing the induced dipoles on polarizable sites, that is, charges on Drude particles, $q_{D,i}$, and opposite charges, $-q_{D,i}$, located on the respective core particles (to which each Drude particle is bonded). Therefore, Thole screening is not applied to the full charge of the core particle q_i, but only to the $-q_{D,i}$ part of it.

The interactions between core charges are subject to the weighting factors set by the *special_bonds* command. The interactions between Drude particles and core charges or non-polarizable atoms are also subject to these weighting factors. The Drude particles inherit the 1-2, 1-3 and 1-4 neighbor relations from their respective cores.

For pair_style *thole*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the example above.

- α (distance units^3)
- damp
- cutoff (distance units)

The last two coefficients are optional. If not specified the global Thole damping parameter or global cutoff specified in the pair_style command are used. In order to specify a cutoff (third argument) a damp parameter (second argument) must also be specified.

For pair style *lj/cut/thole/long*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command.

- ε (energy units)
- σ (length units)
- α (distance units^3)
- damp
- LJ cutoff (distance units)

The last two coefficients are optional and default to the global values from the *pair_style* command line.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.273.4 Mixing, shift, table, tail correction, restart, rRESPA info

The *thole* pair style does not support mixing. Thus, coefficients for all I,J pairs must be specified explicitly.

The *lj/cut/thole/long* pair style does support mixing. Mixed coefficients are defined using

$$\alpha_{ij} = \sqrt{\alpha_i \alpha_j}$$

$$a_{ij} = \frac{1}{2}(a_i + a_j)$$

4.273.5 Restrictions

These pair styles are part of the DRUDE package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair_style should currently not be used with the *charmm dihedral style* if the latter has non-zero 1-4 weighting factors. This is because the *thole* pair style does not know which pairs are 1-4 partners of which dihedrals.

The *lj/cut/thole/long* pair style should be used with a *Kspace solver* like PPPM or Ewald, which is only enabled if LAMMPS was built with the kspace package.

4.273.6 Related commands

fix drude, *fix langevin/drude*, *fix drude/transform*, *compute temp/drude pair_style lj/cut/coul/long*

4.273.7 Default

none

(Noskov) Noskov, Lamoureux and Roux, J Phys Chem B, 109, 6705 (2005).

(Thole) Chem Phys, 59, 341 (1981).

4.274 pair_style threebody/table command

4.274.1 Syntax

```
pair_style style
```

- style = *threebody/table*

4.274.2 Examples

```
pair_style threebody/table
pair_coeff * * spce2.3b type1 type2

pair_style hybrid/overlay table linear 1200 threebody/table
pair_coeff      1 1 table table_CG_CG.txt VOTCA
pair_coeff      * * threebody/table spce.3b type
```

Used in example input scripts:

```
examples/PACKAGES/manybody_table/in.spce
examples/PACKAGES/manybody_table/in.spce2
```

4.274.3 Description

Added in version 2Jun2022.

The *threebody/table* style is a pair style for generic tabulated three-body interactions. It has been developed for (coarse-grained) simulations (of water) with Kernel-based machine learning (ML) potentials (*Scherer2*). As for many other MANYBODY package pair styles the energy of a system is computed as a sum over three-body terms:

$$E = \sum_i \sum_{j \neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk})$$

The summations in the formula are over all neighbors J and K of atom I within a cutoff distance *cut*. In contrast to the Stillinger-Weber potential, all forces are not calculated analytically, but read in from a three-body force/energy table which can be generated with the csg_ml app of VOTCA as available at: https://gitlab.mpcdf.mpg.de/votca/votca.

Only a single pair_coeff command is used with the *threebody/table* style which specifies a threebody potential (".3b") file with parameters for all needed elements. These are then mapped to LAMMPS atom types by specifying N_el additional arguments after the ".3b" filename in the pair_coeff command, where N_el is the number of LAMMPS atom types:

- ".3b" filename
- N_el element names = mapping of threebody elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.3b has three-body values for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.3b Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the ".3b" file. The final C argument maps LAMMPS atom type 4 to the C element in the threebody file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *threebody/table* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

The three-body files have a ".3b" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry specify to the (three-body) cutoff distance and the tabulated three-body interaction. A single entry then contains:

- element 1 (the center atom in a 3-body interaction)
- element 2

- element 3
- cut (distance units)
- filename
- keyword
- style
- N

The parameter *cut* is the (three-body) cutoff distance. When set to 0, no interaction is calculated for this element triplet. The parameters *filename*, *keyword*, *style*, and *N* refer to the tabulated three-body potential.

The tabulation is done on a three-dimensional grid of the two distances r_{ij} and r_{ik} as well as the angle θ_{ijk} which is constructed in the following way. There are two different cases. If element 2 and element 3 are of the same type (e.g. SiCC), the distance r_{ij} is varied in "N" steps from rmin to rmax and the distance r_{ik} is varied from r_{ij} to rmax. This can be done, due to the symmetry of the triplet. If element 2 and element 3 are not of the same type (e.g. SiCSi), there is no additional symmetry and the distance r_{ik} is also varied from rmin to rmax in "N" steps. The angle θ_{ijk} is always varied in "2N" steps from (0.0 + 180.0/(4N)) to (180.0 - 180.0/(4N)). Therefore, the total number of table entries is "M = N * N * (N+1)" for the symmetric (element 2 and element 3 are of the same type) and "M = 2 * N * N * N" for the general case (element 2 and element 3 are not of the same type).

The forces on all three particles I, J, and K of a triplet of this type of three-body interaction potential ($\phi_3(r_{ij}, r_{ik}, \theta_{ijk})$) lie within the plane defined by the three inter-particle distance vectors $\mathbf{r}_{ij}$, $\mathbf{r}_{ik}$, and $\mathbf{r}_{jk}$. This property is used to project the forces onto the inter-particle distance vectors as follows

$$\begin{pmatrix} \mathbf{f}_i \\ \mathbf{f}_j \\ \mathbf{f}_k \end{pmatrix} = \begin{pmatrix} f_{i1} & f_{i2} & 0 \\ f_{j1} & 0 & f_{j2} \\ 0 & f_{k1} & f_{k2} \end{pmatrix} \begin{pmatrix} \mathbf{r}_{ij} \\ \mathbf{r}_{ik} \\ \mathbf{r}_{jk} \end{pmatrix}$$

and then tabulate the 6 force constants f_{i1}, f_{i2}, f_{j1}, f_{j2}, f_{k1}, and f_{k2}, as well as the energy of a triplet e. Due to symmetry reasons, the following relations hold: $f_{i1} = -f_{j1}$, $f_{i2} = -f_{k1}$, and $f_{j2} = -f_{k2}$. As in this pair style the forces are read in directly, a correct MD simulation is also performed in the case that the triplet energies are set to e=0.

The *filename* specifies the file containing the tabulated energy and derivative values of $\phi_3(r_{ij}, r_{ik}, \theta_{ijk})$. The *keyword* then specifies a section of the file. The format of this file is as follows (without the parenthesized comments):

```
# Tabulated three-body potential for spce water (one or more comment or blank lines)

ENTRY1                                                    (keyword is the first text on line)
N 12 rmin 2.55 rmax 3.65                                  (N, rmin, rmax parameters)
                                                          (blank line)
1 2.55 2.55 3.75 -867.212 -611.273 867.212 21386.8 611.273 -21386.8  0.0    (index, r_ij, r_ik, theta, f_i1,
↪f_i2, f_j1, f_j2, f_k1, f_k2, e)
2 2.55 2.55 11.25 -621.539 -411.189 621.539 5035.95 411.189 -5035.95  0.0
...
1872 3.65 3.65 176.25 -0.00215132 -0.00412886 0.00215137 0.00111754 0.00412895 -0.00111757  0.0
```

A section begins with a non-blank line whose first character is not a "#"; blank lines or lines starting with "#" can be used as comments between sections. The first line begins with a keyword which identifies the section. The next line lists (in any order) one or more parameters for the table. Each parameter is a keyword followed by one or more numeric values.

The parameter "N" is required. It should be the same than the parameter "N" of the ".3b" file, otherwise its value is overwritten. "N" determines the number of table entries "M" that follow: "M = N * N * (N+1)" (symmetric triplet, e.g. SiCC) or "M = 2 * N * N * N" (asymmetric triplet, e.g. SiCSi). Therefore "M = 12 * 12 * 13 = 1872" in the above symmetric example. The parameters "rmin" and "rmax" are also required and determine the minimum and maximum of the inter-particle distances r_{ij} and r_{ik}.

Following a blank line, the next M lines of the angular table file list the tabulated values. On each line, the first value is the index from 1 to M, the second value is the distance r_{ij}, the third value is the distance r_{ik}, the fourth value is the angle θ_{ijk}), the next six values are the force constants f_{i1}, f_{i2}, f_{j1}, f_{j2}, f_{k1}, and f_{k2}, and the last value is the energy e.

Note that one three-body potential file can contain many sections, each with a tabulated potential. LAMMPS reads the file section by section until it finds one that matches the specified *keyword* of appropriate section of the ".3b" file.

At the moment, only the *style linear* is allowed and implemented. After reading in the force table, it is internally stored in LAMMPS as a lookup table. For each triplet configuration occurring in the simulation within the cutoff distance, the next nearest tabulated triplet configuration is looked up. No interpolation is done. This allows for a very efficient force calculation with the stored force constants and energies. Due to the know table structure, the lookup can be done efficiently. It has been tested (*Scherer2*) that with a reasonably small bin size, the accuracy and speed is comparable to that of a Stillinger-Weber potential with tabulated three-body interactions (*pair_style sw/angle/table*) while the table format of this pair style allows for more flexible three-body interactions.

As for the Stillinger-Weber potential, the three-body potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries.

For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify threebody parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

As annotated above, the first element in the entry is the center atom in a three-body interaction. Thus an entry for SiCC means a Si atom with 2 C atoms as neighbors. The tabulated three-body forces can in principle be specific to the three elements of the configuration. However, the user must ensure that it makes physically sense. E.g., the tabulated three-body forces for the entries CSiC and CCSi should be the same exchanging r_{ij} with r_{ik}, f_{j1} with f_{k1}, and f_{j2} with f_{k2}.

Additional input files and reference data can be found at: https://gitlab.mpcdf.mpg.de/votca/votca/-/tree/master/csg-tutorials/ml

---

4.274.4 Mixing, shift, table, tail correction, restart, rRESPA info

As all interactions are tabulated, no mixing is performed.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.274.5 Restrictions

This pair style is part of the MANYBODY package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on" for pair interactions.

4.274.6 Related commands

pair_coeff, *pair sw/angle/table*

(Scherer2) C. Scherer, R. Scheid, D. Andrienko, and T. Bereau, J. Chem. Theor. Comp. 16, 3194-3204 (2020).

4.275 pair_style tracker command

4.275.1 Syntax

```
pair_style tracker fix_ID N keyword values attribute1 attribute2 ...
```

- fix_ID = ID of associated internal fix to store data
- N = prepare data for output every this many timesteps
- zero or more keywords may be appended
- keyword = *finite* or *time/min* or *type/include*

  finite value = none
    pair style uses atomic diameters to identify contacts
  time/min value = T
    T = minimum number of timesteps of interaction
  type/include value = list1 list2
    list1,list2 = separate lists of types (see below)

- one or more attributes may be appended

```
possible attributes = id1 id2 time/created time/broken time/total
                      r/min r/ave x y z
```

```
id1, id2 = IDs of the two atoms in each pair interaction
time/created = the timestep that the two atoms began interacting
time/broken = the timestep that the two atoms stopped interacting
time/total = the total number of timesteps the two atoms interacted
r/min = the minimum radial distance between the two atoms during the interaction (distance units)
r/ave = the average radial distance between the two atoms during the interaction (distance units)
x, y, z = the center of mass position of the two atoms when they stopped interacting (distance units)
```

4.275.2 Examples

```
pair_style hybrid/overlay tracker myfix 1000 id1 id2 type/include 1 * type/include 2 3,4  lj/cut 2.5
pair_coeff 1 1 tracker 2.0

pair_style hybrid/overlay tracker myfix 1000 finite x y z time/min 100 granular
pair_coeff * * tracker

dump 1 all local 1000 dump.local f_myfix[1] f_myfix[2] f_myfix[3]
dump_modify 1 write_header no
```

4.275.3 Description

Style *tracker* monitors information about pairwise interactions. It does not calculate any forces on atoms. *Pair hybrid/overlay* can be used to combine this pair style with any other pair style, as shown in the examples above.

At each timestep, if two neighboring atoms move beyond the interaction cutoff, pairwise data is processed and transferred to an internal fix labeled *fix_ID*. This allows the local data to be accessed by other LAMMPS commands. Additional filters can be applied using the *time/min* or *type/include* keywords described below. Note that this is the interaction cutoff defined by this pair style, not the short-range cutoff defined by the pair style that is calculating forces on atoms.

Following any optional keyword/value arguments, a list of one or more attributes is specified. These include the IDs of the two atoms in the pair. The other attributes for the pair of atoms are the duration of time they were "interacting" or at the point in time they started or stopped interacting. In this context, "interacting" means the time window during which the two atoms were closer than the interaction cutoff distance. The attributes for time/* refer to timesteps.

Data is continuously accumulated by the internal fix over intervals of *N* timesteps. At the end of each interval, all of the saved accumulated data is deleted to make room for new data. Individual datum may therefore persist anywhere between *1* to *N* timesteps depending on when they are saved. This data can be accessed using the *fix_ID* and a *dump local* command. To ensure all data is output, the dump frequency should correspond to the same interval of *N* timesteps. A dump frequency of an integer multiple of *N* can be used to regularly output a sample of the accumulated data.

The following optional keywords may be used.

If the *finite* keyword is not used, the following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutoff (distance units)

If the *finite* keyword is used, there are no additional coefficients to set for each pair of atom types via the *pair_coeff* command. Interaction cutoffs are instead calculated based on the diameter of finite particles. However you must still use the *pair_coeff* for all atom types. For example the command

```
pair_coeff * *
```

should be used.

The *time/min* keyword sets a minimum amount of time that an interaction must persist to be included. This setting can be used to censor short-lived interactions.

The *type/include* keyword filters interactions based on the types of the two atoms. Data is only saved for interactions between atoms whose two atom types appear in *list1* and *list2*. Atom type 1 must be in list1 and atom type 2 in list2. Or vice versa.

Each type list consists of a series of type ranges separated by commas. Each range can be specified as a single numeric value, or a wildcard asterisk can be used to specify a range of values. This takes the form "*" or "*n" or "n*" or "m*n". For example, if M = the number of atom types, then an asterisk with no numeric values means all types from 1 to M. A leading asterisk means all types from 1 to n (inclusive). A trailing asterisk means all types from n to M (inclusive). A middle asterisk means all types from m to n (inclusive). Note that the *type/include* keyword can be specified multiple times.

4.275.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the cutoff coefficient and cutoff distance for this pair style can be mixed. The cutoff is always mixed via a *geometric* rule. The cutoff is mixed according to the pair_modify mix value. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

The *pair_modify* shift, table, and tail options are not relevant for this pair style.

The accumulated data is not written to restart files and should be output before a restart file is written to avoid missing data.

The internal fix calculates a local vector or local array depending on the number of input values. The length of the vector or number of rows in the array is the number of recorded, lost interactions. If a single input is specified, a local vector is produced. If two or more inputs are specified, a local array is produced where the number of columns = the number of inputs. The vector or array can be accessed by any command that uses local values from a compute as input. See the *Howto output* page for an overview of LAMMPS output options.

The vector or array will be floating point values that correspond to the specified attribute.

4.275.5 Restrictions

This pair style is part of the MISC package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style is currently incompatible with granular pair styles that extend beyond the contact (e.g. JKR and DMT).

4.275.6 Related commands

4.275.7 Default

none

4.276 pair_style tri/lj command

4.276.1 Syntax

```
pair_style tri/lj cutoff
```

cutoff = global cutoff for interactions (distance units)

4.276.2 Examples

```
pair_style tri/lj 3.0
pair_coeff * * 1.0 1.0
pair_coeff 1 1 1.0 1.5 2.5
```

4.276.3 Description

Style *tri/lj* treats particles which are triangles as a set of small spherical particles that tile the triangle surface as explained below. Interactions between two triangles, each with N1 and N2 spherical particles, are calculated as the pairwise sum of N1*N2 Lennard-Jones interactions. Interactions between a triangle with N spherical particles and a point particle are treated as the pairwise sum of N Lennard-Jones interactions. See the *pair_style lj/cut* doc page for the definition of Lennard-Jones interactions.

The cutoff distance for an interaction between 2 triangles, or between a triangle and a point particle, is calculated from the position of the triangle (its centroid), not between pairs of individual spheres comprising the triangle. Thus an interaction is either calculated in its entirety or not at all.

The set of non-overlapping spherical particles that represent a triangle, for purposes of this pair style, are generated in the following manner. Assume the triangle is of type I, and sigma_II has been specified. We want a set of spheres with centers in the plane of the triangle, none of them larger in diameter than sigma_II, which completely cover the triangle's area, but with minimal overlap and a minimal total number of spheres. This is done in a recursive manner. Place a sphere at the centroid of the original triangle. Calculate what diameter it must have to just cover all 3 corner points of the triangle. If that diameter is equal to or smaller than sigma_II, then include a sphere of the calculated diameter in the set of covering spheres. It the diameter is larger than sigma_II, then split the triangle into 2 triangles by bisecting its longest side. Repeat the process on each sub-triangle, recursing as far as needed to generate a set of covering spheres. When finished, the original criteria are met, and the set of covering spheres should be near minimal in number and overlap, at least for input triangles with a reasonable aspect-ratio.

The LJ interaction between 2 spheres on different triangles of types I,J is computed with an arithmetic mixing of the sigma values of the 2 spheres and using the specified epsilon value for I,J atom types. Note that because the sigma values for triangles spheres is computed using only sigma_II values, specific to the triangles's type, this means that any specified sigma_IJ values (for I != J) are effectively ignored.

For style *tri/lj*, the following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- epsilon (energy units)
- sigma (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff is used.

4.276.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for all of this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.276.5 Restrictions

This style is part of the ASPHERE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

Defining particles to be triangles so they participate in tri/tri or tri/particle interactions requires the use the *atom_style tri* command.

4.276.6 Related commands

pair_coeff , *pair_style line/lj*

4.276.7 Default

none

4.277 pair_style uf3 command

Accelerator Variants: *uf3/kk*

4.277.1 Syntax

```
pair_style style BodyFlag
```

- style = *uf3* or *uf3/kk*

  ```
  BodyFlag = Indicates whether to calculate only 2-body or 2 and 3-body interactions. Possible␣
  ↪values: 2 or 3
  ```

4.277.2 Examples

```
pair_style uf3 3
pair_coeff * * Nb.uf3 Nb

pair_style uf3 2
pair_coeff * * NbSn.uf3 Nb Sn

pair_style uf3 3
pair_coeff * * NbSn.uf3 Nb Sn
```

4.277.3 Description

Added in version 27June2024.

The *uf3* style computes the *Ultra-Fast Force Fields (UF3)* potential, a machine-learning interatomic potential. In UF3, the total energy of the system is defined via two- and three-body interactions:

$$E = \sum_{i,j} V_2(r_{ij}) + \sum_{i,j,k} V_3(r_{ij}, r_{ik}, r_{jk})$$

$$V_2(r_{ij}) = \sum_{n=0}^{N} c_n B_n(r_{ij})$$

$$V_3(r_{ij}, r_{ik}, r_{jk}) = \sum_{l=0}^{N_l} \sum_{m=0}^{N_m} \sum_{n=0}^{N_n} c_{l,m,n} B_l(r_{ij}) B_m(r_{ik}) B_n(r_{jk})$$

where $V_2(r_{ij})$ and $V_3(r_{ij}, r_{ik}, r_{jk})$ are the two- and three-body interactions, respectively. For the two-body the summation is over all neighbors J and for the three-body the summation is over all neighbors J and K of atom I within a cutoff distance determined from the potential files. $B_n(r_{ij})$ are the cubic b-spline basis, c_n and $c_{l,m,n}$ are the machine-learned interaction parameters and N, N_l, N_m, and N_n denote the number of basis functions per spline or tensor spline dimension.

With *uf3* style only a single pair_coeff command is used to indicate the UF3 LAMMPS potential file containing all the two- and three-body interactions followed by N additional arguments specifying the mapping of UF3 elements to LAMMPS atom types, where N is the number of LAMMPS atom types:

- UF3 LAMMPS potential file
- N elements names = mapping of UF3 elements to atom types

As an example, if a LAMMPS simulation contains 2 atom types (elements 'A' and 'B'), the pair_coeff command will be:

```
pair_style uf3 3
pair_coeff * * AB.uf3 A B
```

The AB.uf3 file should contain all two-body (A-A, A-B, B-B) and three-body (A-A-A, A-A-B, A-B-B, B-A-A, B-A-B, B-B-B).

If a value of "2" is specified in the pair_style uf3 command, only the two-body potentials are needed. For 3-body interaction the first atom type is the central atom. We recommend using the generate_uf3_lammps_pots.py script (found here) for generating the UF3 LAMMPS potential file from the UF3 JSON potentials.

UF3 LAMMPS potential file in the *potentials* directory of the LAMMPS distribution have a ".uf3" suffix. The interaction block in UF3 LAMMPS potential file should start with #UF3 POT and end with # characters. Following shows the format of a generic 2-body and 3-body potential block in UF3 LAMMPS potential file-

```
#UF3 POT UNITS: units DATE: POT_GEN_DATE AUTHOR: AUTHOR_NAME CITATION: CITE
2B ELEMENT1 ELEMENT2 LEADING_TRIM TRAILING_TRIM
Rij_CUTOFF NUM_OF_KNOTS
BSPLINE_KNOTS
NUM_OF_COEFF
COEFF
#
#UF3 POT UNITS: units DATE: POT_GEN_DATE AUTHOR: AUTHOR_NAME CITATION: CITE
3B ELEMENT1 ELEMENT2 ELEMENT3 LEADING_TRIM TRAILING_TRIM
Rjk_CUTOFF Rik_CUTOFF Rij_CUTOFF NUM_OF_KNOTS_JK NUM_OF_KNOTS_IK NUM_
↪OF_KNOTS_IJ
BSPLINE_KNOTS_FOR_JK
BSPLINE_KNOTS_FOR_IK
BSPLINE_KNOTS_FOR_IJ
SHAPE_OF_COEFF_MATRIX[I][J][K]
COEFF_MATRIX[0][0][K]
COEFF_MATRIX[0][1][K]
COEFF_MATRIX[0][2][K]
.
.
.
COEFF_MATRIX[1][0][K]
COEFF_MATRIX[1][1][K]
COEFF_MATRIX[1][2][K]
.
.
.
#
```

The second line indicates whether the block contains data for 2-body (2B) or 3-body (3B) interaction. This is followed by element combination interaction, LEADING_TRIM and TRAILING_TRIM number on the same line. The current implementation is only tested for LEADING_TRIM=0 and TRAILING_TRIM=3. If other values are used LAMMPS is terminated after issuing an error message. The Rij_CUTOFF sets the 2-body cutoff for the interaction described by the potential block. NUM_OF_KNOTS is the number of knots (or the length of the knot vector) present on the very next line. The BSPLINE_KNOTS line should contain all the knots in ascending order. NUM_OF_COEFF is the number of coefficients in the COEFF line. All the numbers in the BSPLINE_KNOTS and COEFF line should be space-separated. Similar to the 2-body potential block, the third line sets the cutoffs and length of the knots. The cutoff distance between atom-type I and J is Rij_CUTOFF, atom-type I and K is Rik_CUTOFF and between J and K is Rjk_CUTOFF.

Note

The current implementation only works for UF3 potentials with cutoff distances for 3-body interactions that follows 2Rij_CUTOFF=2Rik_CUTOFF=Rjk_CUTOFF relation.

The BSPLINE_KNOTS_FOR_JK, BSPLINE_KNOTS_FOR_IK, and BSPLINE_KNOTS_FOR_IJ lines (note the order) contain the knots in increasing order for atoms J and K, I and K, and atoms I and J respectively. The number of knots is defined by the NUM_OF_KNOTS_* characters in the previous line. The shape of the coefficient matrix is defined on the SHAPE_OF_COEFF_MATRIX[I][J][K] line followed by the columns of the coefficient matrix, one per line, as shown above. For example, if the coefficient matrix has the shape of 8x8x13, then SHAPE_OF_COEFF_MATRIX[I][J][K] will be 8 8 13 followed by 64 (8x8) lines each containing 13 coefficients separated by space.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.277.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.277.5 Restrictions

The 'uf3' pair style is part of the ML-UF3 package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires the *newton* setting to be "on".

The UF3 LAMMPS potential file provided with LAMMPS (see the potentials directory) are parameterized for metal *units*.

The single() function of 'uf3' pair style only return the 2-body interaction energy.

4.277.6 Related commands

pair_coeff

4.277.7 Default

none

(Xie23) Xie, S.R., Rupp, M. & Hennig, R.G. Ultra-fast interpretable machine-learning potentials. npj Comput Mater 9, 162 (2023). https://doi.org/10.1038/s41524-023-01092-7

4.278 pair_style ufm command

Accelerator Variants: *ufm/gpu*, *ufm/omp*, *ufm/opt*

4.278.1 Syntax

```
pair_style ufm cutoff
```

- cutoff = global cutoff for *ufm* interactions (distance units)

4.278.2 Examples

```
pair_style ufm 4.0
pair_coeff 1 1 100.0 1.0 2.5
pair_coeff * * 100.0 1.0

pair_style ufm 4.0
pair_coeff * * 10.0 1.0
variable prefactor equal ramp(10,100)
fix 1 all adapt 1 pair ufm epsilon * * v_prefactor
```

4.278.3 Description

Style *ufm* computes pairwise interactions using the Uhlenbeck-Ford model (UFM) potential *(Paula Leite2016)* which is given by

$$E = -\varepsilon \ln\left[1 - \exp\left(-r^2/\sigma^2\right)\right] \qquad r < r_c$$
$$\varepsilon = p\, k_B\, T$$

where r_c is the cutoff, σ is a distance-scale and ε is an energy-scale, i.e., a product of Boltzmann constant k_B, temperature T and the Uhlenbeck-Ford p-parameter which is responsible to control the softness of the interactions *(Paula Leite2017)*. This model is useful as a reference system for fluid-phase free-energy calculations *(Paula Leite2016)*.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε (energy units)
- σ (distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global *ufm* cutoff is used.

The *fix adapt* command can be used to vary epsilon and sigma for this pair style over the course of a simulation, in which case pair_coeff settings for epsilon and sigma must still be specified, but will be overridden. For example these commands will vary the prefactor epsilon for all pairwise interactions from 10.0 at the beginning to 100.0 at the end of a run:

```
variable prefactor equal ramp(10,100)
fix 1 all adapt 1 pair ufm epsilon * * v_prefactor
```

Note

The thermodynamic integration procedure can be performed with this potential using *fix adapt*. This command will rescale the force on each atom by varying a scale variable, which always starts with value 1.0. The syntax is the same described above, however, changing epsilon to scale. A detailed explanation of how to use this command and perform nonequilibrium thermodynamic integration in LAMMPS is given in the paper by *(Freitas)*.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.278.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style support the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table and tail are not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.278.5 Restrictions

This pair style is part of the EXTRA-PAIR package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

4.278.6 Related commands

pair_coeff, *fix adapt*

4.278.7 Default

none

(Paula Leite2017) Paula Leite, Santos-Florez, and de Koning, Phys Rev E, 96, 32115 (2017).

(Paula Leite2016) Paula Leite , Freitas, Azevedo, and de Koning, J Chem Phys, 126, 044509 (2016).

(Freitas) Freitas, Asta, and de Koning, Computational Materials Science, 112, 333 (2016).

4.279 pair_style vashishta command

Accelerator Variants: *vashishta/gpu*, *vashishta/omp*, *vashishta/kk*

4.280 pair_style vashishta/table command

Accelerator Variants: *vashishta/table/omp*

4.280.1 Syntax

```
pair_style style args
```

- style = *vashishta* or *vashishta/table*
- args = list of arguments for a particular style

```
vashishta args = none
vashishta/table args = Ntable cutinner
  Ntable = # of tabulation points
  cutinner = tablulate from cutinner to cutoff
```

4.280.2 Examples

```
pair_style vashishta
pair_coeff * * SiC.vashishta Si C

pair_style vashishta/table 100000 0.2
pair_coeff * * SiC.vashishta Si C
```

4.280.3 Description

The *vashishta* and *vashishta/table* styles compute the combined 2-body and 3-body family of potentials developed in the group of Priya Vashishta and collaborators. By combining repulsive, screened Coulombic, screened charge-dipole, and dispersion interactions with a bond-angle energy based on the Stillinger-Weber potential, this potential has been used to describe a variety of inorganic compounds, including SiO2 *Vashishta1990*, SiC *Vashishta2007*, and InP *Branicio2009*.

The potential for the energy U of a system of atoms is

$$U = \sum_{i}^{N} \sum_{j>i}^{N} U_{ij}^{(2)}(r_{ij}) + \sum_{i}^{N} \sum_{j \neq i}^{N} \sum_{k>j, k \neq i}^{N} U_{ijk}^{(3)}(r_{ij}, r_{ik}, \theta_{ijk})$$

$$U_{ij}^{(2)}(r) = \frac{H_{ij}}{r^{\eta_{ij}}} + \frac{Z_i Z_j}{r} \exp(-r/\lambda_{1,ij}) - \frac{D_{ij}}{r^4} \exp(-r/\lambda_{4,ij}) - \frac{W_{ij}}{r^6}, r < r_{c,ij}$$

$$U_{ijk}^{(3)}(r_{ij}, r_{ik}, \theta_{ijk}) = B_{ijk} \frac{\left[\cos\theta_{ijk} - \cos\theta_{0ijk}\right]^2}{1 + C_{ijk}\left[\cos\theta_{ijk} - \cos\theta_{0ijk}\right]^2} \times \exp\left(\frac{\gamma_{ij}}{r_{ij} - r_{0,ij}}\right) \exp\left(\frac{\gamma_{ik}}{r_{ik} - r_{0,ik}}\right), r_{ij} < r_{0,ij}, r_{ik} < r_{0,ik}$$

where we follow the notation used in *Branicio2009*. U^2 is a two-body term and U3 is a three-body term. The summation over two-body terms is over all neighbors J within a cutoff distance = r_c. The twobody terms are shifted and tilted by a linear function so that the energy and force are both zero at r_c. The summation over three-body terms is over all neighbors i and k within a cut-off distance = r_0, where the exponential screening function becomes zero.

The *vashishta* style computes these formulas analytically. The *vashishta/table* style tabulates the analytic values for *Ntable* points from cutinner to the cutoff of the potential. The points are equally spaced in R^2 space from cutinner^2 to cutoff^2. For the two-body term in the above equation, a linear interpolation for each pairwise distance between adjacent points in the table. In practice the tabulated version can run 3-5x faster than the analytic version with moderate to little loss of accuracy for Ntable values between 10000 and 1000000. It is not recommended to use less than 5000 tabulation points.

Only a single pair_coeff command is used with either style which specifies a Vashishta potential file with parameters for all needed elements. These are mapped to LAMMPS atom types by specifying N additional arguments after the filename in the pair_coeff command, where N is the number of LAMMPS atom types:

- filename
- N element names = mapping of Vashishta elements to atom types

See the *pair_coeff* page for alternate ways to specify the path for the potential file.

As an example, imagine a file SiC.vashishta has parameters for Si and C. If your LAMMPS simulation has 4 atoms types and you want the first 3 to be Si, and the fourth to be C, you would use the following pair_coeff command:

```
pair_coeff * * SiC.vashishta Si Si Si C
```

The first 2 arguments must be * * so as to span all LAMMPS atom types. The first three Si arguments map LAMMPS atom types 1,2,3 to the Si element in the file. The final C argument maps LAMMPS atom type 4 to the C element in the file. If a mapping value is specified as NULL, the mapping is not performed. This can be used when a *vashishta* potential is used as part of the *hybrid* pair style. The NULL values are placeholders for atom types that will be used with other potentials.

Vashishta files in the *potentials* directory of the LAMMPS distribution have a ".vashishta" suffix. Lines that are not blank or comments (starting with #) define parameters for a triplet of elements. The parameters in a single entry correspond to the two-body and three-body coefficients in the formulae above:

- element 1 (the center atom in a 3-body interaction)
- element 2

- element 3
- H (energy units)
- η
- Z_i (electron charge units)
- Z_j (electron charge units)
- λ_1 (distance units)
- D (energy units)
- λ_4 (distance units)
- W (energy units)
- r_c (distance units)
- B (energy units)
- γ
- r_0 (distance units)
- C
- $\cos \theta_0$

The non-annotated parameters are unitless. The Vashishta potential file must contain entries for all the elements listed in the pair_coeff command. It can also contain entries for additional elements not being used in a particular simulation; LAMMPS ignores those entries. For a single-element simulation, only a single entry is required (e.g. SiSiSi). For a two-element simulation, the file must contain 8 entries (for SiSiSi, SiSiC, SiCSi, SiCC, CSiSi, CSiC, CCSi, CCC), that specify parameters for all permutations of the two elements interacting in three-body configurations. Thus for 3 elements, 27 entries would be required, etc.

Depending on the particular version of the Vashishta potential, the values of these parameters may be keyed to the identities of zero, one, two, or three elements. In order to make the input file format unambiguous, general, and simple to code, LAMMPS uses a slightly confusing method for specifying parameters. All parameters are divided into two classes: two-body and three-body. Two-body and three-body parameters are handled differently, as described below. The two-body parameters are H, η, λ_1, D, λ_4, W, r_c, γ, and r_0. They appear in the above formulae with two subscripts. The parameters Z_i and Z_j are also classified as two-body parameters, even though they only have 1 subscript. The three-body parameters are B, C, $\cos \theta_0$. They appear in the above formulae with three subscripts. Two-body and three-body parameters are handled differently, as described below.

The first element in each entry is the center atom in a three-body interaction, while the second and third elements are two neighbor atoms. Three-body parameters for a central atom I and two neighbors J and K are taken from the IJK entry. Note that even though three-body parameters do not depend on the order of J and K, LAMMPS stores three-body parameters for both IJK and IKJ. The user must ensure that these values are equal. Two-body parameters for an atom I interacting with atom J are taken from the IJJ entry, where the second and third elements are the same. Thus the two-body parameters for Si interacting with C come from the SiCC entry. Note that even though two-body parameters (except possibly gamma and r0 in U3) do not depend on the order of the two elements, LAMMPS will get the Si-C value from the SiCC entry and the C-Si value from the CSiSi entry. The user must ensure that these values are equal. Two-body parameters appearing in entries where the second and third elements are different are stored but never used. It is good practice to enter zero for these values. Note that the three-body function U3 above contains the two-body parameters γ and r_0. So U3 for a central C atom bonded to an Si atom and a second C atom will take three-body parameters from the CSiC entry, but two-body parameters from the CCC and CSiSi entries.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages*

page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.280.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, where types I and J correspond to two different element types, mixing is performed by LAMMPS as described above from values in the potential file.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style does not write its information to *binary restart files*, since it is stored in potential files. Thus, you need to re-specify the pair_style and pair_coeff commands in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.280.5 Restrictions

These pair styles are part of the MANYBODY package. They are only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

These pair styles requires the *newton* setting to be "on" for pair interactions.

The Vashishta potential files provided with LAMMPS (see the potentials directory) are parameterized for metal *units*. You can use the Vashishta potential with any LAMMPS units, but you would need to create your own potential file with coefficients listed in the appropriate units if your simulation does not use "metal" units.

4.280.6 Related commands

pair_coeff

4.280.7 Default

none

(Vashishta1990) P. Vashishta, R. K. Kalia, J. P. Rino, Phys. Rev. B 41, 12197 (1990).

(Vashishta2007) P. Vashishta, R. K. Kalia, A. Nakano, J. P. Rino. J. Appl. Phys. 101, 103515 (2007).

(Branicio2009) Branicio, Rino, Gan and Tsuzuki, J. Phys Condensed Matter 21 (2009) 095002

4.281 pair_style wf/cut command

4.281.1 Syntax

```
pair_style wf/cut cutoff
```

- cutoff = cutoff for wf interactions (distance units)

4.281.2 Examples

```
pair_style      wf/cut 2.0
pair_coeff      1 1 1.0 1.0 1 1  2.0
```

4.281.3 Description

The *wf/cut* (Wang-Frenkel) style computes LJ-like potentials as described in *Wang2020*. This potential is by construction finite ranged and it vanishes quadratically at the cutoff distance, avoiding truncation, shifting, interpolation and other typical procedures with the LJ potential. The *wf/cut* can be used when a typical short-ranged potential with attraction is required. The potential is given by which is given by:

$$\phi(r) = \varepsilon\alpha \left(\left[\frac{\sigma}{r}\right]^{2\mu} - 1 \right) \left(\left[\frac{r_c}{r}\right]^{2\mu} - 1 \right)^{2\nu}$$

with

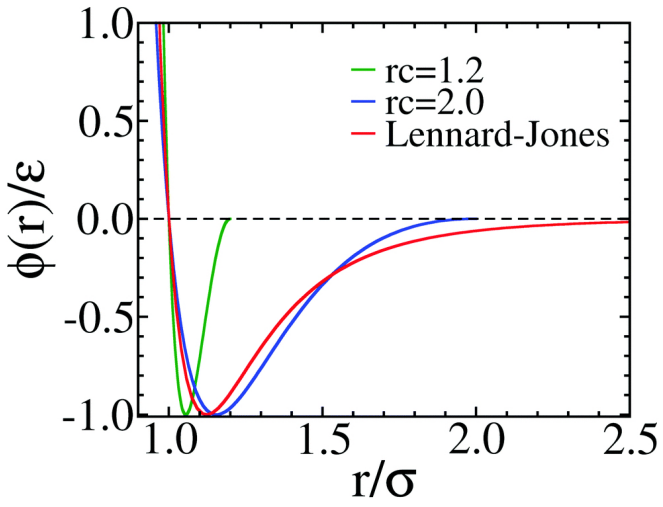
$$\alpha = 2\nu \left(\frac{r_c}{\sigma}\right)^{2\mu} \left[\frac{1+2\nu}{2\nu \left[(r_c/\sigma)^{2\mu} - 1\right]} \right]^{2\nu+1}$$

and

$$r_{min} = r_c \left[\frac{1+2\nu}{1+2\nu(r_c/\sigma)^{2\nu}} \right]^{1/2\nu}$$

r_c is the cutoff.

Comparison of the non-truncated Lennard-Jones 12-6 potential (red curve), and the WF potentials with $\mu = 1$ and $\nu = 1$ are shown in the figure below. The blue curve has $r_c = 2.0$ and the green curve has $r_c = 1.2$ and can be used to describe colloidal interactions.



The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- ε (energy units)
- σ (distance units)
- ν
- μ
- r_c (distance units)

The last coefficient is optional. If not specified, the global cutoff given in the pair_style command is used. The exponents ν and μ are positive integers, usually set to 1. There is usually little to be gained by choosing other values of ν and μ (See discussion in *Wang2020*)

Mixing, shift, table, tail correction, restart, rRESPA info:

This pair style does not support the *pair_modify* mixing and table options.

The *pair_modify* tail and shift options are not relevant for this pair style as it goes to zero at the cut-off radius.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style does not support the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command.

4.281.4 Restrictions

This pair style can only be used if LAMMPS was built with the EXTRA-PAIR package. See the *Build package* doc page for more info.

4.281.5 Related commands

pair_coeff

Default: none

(Wang2020) X. Wang, S. Ramirez-Hinestrosa, J. Dobnikar, and D. Frenkel, Phys. Chem. Chem. Phys. 22, 10624 (2020).

4.282 pair_style ylz command

4.282.1 Syntax

```
pair_style ylz cutoff
```

- cutoff = global cutoff for interactions (distance units)

4.282.2 Examples

```
pair_style   ylz  2.6
pair_coeff   *  *  1.0  1.0  4  3  0.0  2.6
```

4.282.3 Description

Added in version 3Nov2022.

The *ylz* (Yuan-Li-Zhang) style computes an anisotropic interaction between pairs of coarse-grained particles considering the relative particle orientations. This potential was originally developed as a particle-based solvent-free model for biological membranes *(Yuan2010a)*. Unlike *pair_style gayberne*, whose orientation dependence is strictly derived from the closest distance between two ellipsoidal rigid bodies, the orientation-dependence of this pair style is mathematically defined such that the particles can self-assemble into one-particle-thick fluid membranes. The potential of this pair style is described by:

$$U(\mathbf{r}_{ij}, \mathbf{n}_i, \mathbf{n}_j) = \begin{cases} u_R(r) + \left[1 - \phi(\hat{\mathbf{r}}_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right] \varepsilon, & r < r_{min} \\ u_A(r)\phi(\hat{\mathbf{r}}_{ij}, \mathbf{n}_i, \mathbf{n}_j), & r_{min} < r < r_c \end{cases}$$

$$\phi(\hat{\mathbf{r}}_{ij}, \mathbf{n}_i, \mathbf{n}_j) = 1 + \left[\mu(a(\hat{\mathbf{r}}_{ij}, \mathbf{n}_i, \mathbf{n}_j) - 1)\right]$$

$$a(\hat{\mathbf{r}}_{ij}, \mathbf{n}_i, \mathbf{n}_j) = (\mathbf{n}_i \times \hat{\mathbf{r}}_{ij}) \cdot (\mathbf{n}_j \times \hat{\mathbf{r}}_{ij}) + \beta(\mathbf{n}_i - \mathbf{n}_j) \cdot \hat{\mathbf{r}}_{ij} - \beta^2$$

$$u_R(r) = \varepsilon \left[\left(\frac{r_{min}}{r}\right)^4 - 2\left(\frac{r_{min}}{r}\right)^2\right]$$

$$u_A(r) = -\varepsilon \, cos^{2\zeta}\left[\frac{\pi}{2}\frac{(r - r_{min})}{(r_c - r_{min})}\right]$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the center position vectors of particles i and j, respectively, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ is the inter-particle distance vector, $r = |\mathbf{r}_{ij}|$ and $\hat{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/r$. The unit vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ represent the axes of symmetry of particles i and j, respectively, u_R and u_A are the repulsive and attractive potentials, ϕ is an angular function which depends on the relative orientation between pair particles, μ is the parameter related to the bending rigidity of the membrane, β is the parameter related to the spontaneous curvature, and ε is the energy unit, respectively. The ζ controls the slope of the attractive branch and hence the diffusivity of the particles in the in-plane direction of the membrane. r_c is the cutoff radius, r_{min} is the distance which minimizes the potential energy $u_A(r)$ and $r_{min} = 2^{1/6}\sigma$, where σ is the length unit.

This pair style is suited for solvent-free coarse-grained simulations of biological systems involving lipid bilayer membranes, such as vesicle shape transformations *(Yuan2010b)*, nanoparticle endocytosis *(Huang)*, modeling of red blood cell membranes *(Fu)*, *(Appshaw)*, and modeling of cell elasticity *(Becton)*.

Use of this pair style requires the NVE, NVT, or NPT fixes with the *asphere* extension (e.g. *fix nve/asphere*) in order to integrate particle rotation. Additionally, *atom_style ellipsoid* should be used since it defines the rotational state of each particle.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- ε = well depth (energy units)
- σ = minimum effective particle radii (distance units)
- ζ = tuning parameter for the slope of the attractive branch

- μ = parameter related to bending rigidity
- β = parameter related to the spontaneous curvature
- cutoff (distance units)

The last coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used.

4.282.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the epsilon and sigma coefficients and cutoff distance for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.282.5 Restrictions

The *ylz* style is part of the ASPHERE package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires that atoms store torque and a quaternion to represent their orientation, as defined by the *atom_style*. It also requires they store a per-atom *shape*. The particles cannot store a per-particle diameter. To avoid being mistakenly considered as point particles, the shape parameters ought to be non-spherical, like [1 0.99 0.99]. Unlike the *resquared* pair style for which the shape directly determines the mathematical expressions of the potential, the shape parameters for this pair style is only involved in the computation of the moment of inertia and thus only influences the rotational dynamics of individual particles.

This pair style requires that **all** atoms are ellipsoids as defined by the *atom_style ellipsoid* command.

4.282.6 Related commands

pair_coeff, *fix nve/asphere*, *compute temp/asphere*, *pair_style resquared*, *pair_style gayberne*

4.282.7 Default

none

(Yuan2010a) Yuan, Huang, Li, Lykotrafitis, Zhang, Phys. Rev. E, 82, 011905(2010).

(Yuan2010b) Yuan, Huang, Zhang, Soft. Matter, 6, 4571(2010).

(Huang) Huang, Zhang, Yuan, Gao, Zhang, Nano Lett. 13, 4546(2013).

(Fu) Fu, Peng, Yuan, Kfoury, Young, Comput. Phys. Commun, 210, 193-203(2017).

(Appshaw) Appshaw, Seddon, Hanna, Soft. Matter,18, 1747(2022).

(Becton) Becton, Averett, Wang, Biomech. Model. Mechanobiology, 18, 425-433(2019).

4.283 pair_style yukawa command

Accelerator Variants: *yukawa/gpu*, *yukawa/omp*, *yukawa/kk*

4.283.1 Syntax

```
pair_style yukawa kappa cutoff
```

- kappa = screening length (inverse distance units)
- cutoff = global cutoff for Yukawa interactions (distance units)

4.283.2 Examples

```
pair_style yukawa 2.0 2.5
pair_coeff 1 1 100.0 2.3
pair_coeff * * 100.0
```

4.283.3 Description

Style *yukawa* computes pairwise interactions with the formula

$$E = A\frac{e^{-\kappa r}}{r} \qquad r < r_c$$

r_c is the cutoff.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- A (energy*distance units)
- cutoff (distance units)

The last coefficient is optional. If not specified, the global yukawa cutoff is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.283.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A coefficient and cutoff distance for this pair style can be mixed. A is an energy value mixed like a LJ epsilon. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.283.5 Restrictions

none

4.283.6 Related commands

pair_coeff

4.283.7 Default

none

4.284 pair_style yukawa/colloid command

Accelerator Variants: *yukawa/colloid/gpu*, *yukawa/colloid/kk*, *yukawa/colloid/omp*

4.284.1 Syntax

```
pair_style yukawa/colloid kappa cutoff
```

- kappa = screening length (inverse distance units)
- cutoff = global cutoff for colloidal Yukawa interactions (distance units)

4.284.2 Examples

```
pair_style yukawa/colloid 2.0 2.5
pair_coeff 1 1 100.0 2.3
pair_coeff * * 100.0
```

4.284.3 Description

Style *yukawa/colloid* computes pairwise interactions with the formula

$$E = \frac{A}{\kappa} e^{-\kappa(r-(r_i+r_j))} \qquad r < r_c$$

where r_i and r_j are the radii of the two particles and r_c is the cutoff.

In contrast to *pair_style yukawa*, this functional form arises from the Coulombic interaction between two colloid particles, screened due to the presence of an electrolyte, see the book by *Safran* for a derivation in the context of DLVO theory. *Pair_style yukawa* is a screened Coulombic potential between two point-charges and uses no such approximation.

This potential applies to nearby particle pairs for which the Derjagin approximation holds, meaning $h << r_i + r_j$, where h is the surface-to-surface separation of the two particles.

When used in combination with *pair_style colloid*, the two terms become the so-called DLVO potential, which combines electrostatic repulsion and van der Waals attraction.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- A (energy/distance units)
- cutoff (distance units)

The prefactor A is determined from the relationship between surface charge and surface potential due to the presence of electrolyte. Note that the A for this potential style has different units than the A used in *pair_style yukawa*. For low surface potentials, i.e. less than about 25 mV, A can be written as:

$$A = 2\pi R \varepsilon \varepsilon_0 \kappa \psi^2$$

where

- R = colloid radius (distance units)
- ε_0 = permittivity of free space (charge^2/energy/distance units)
- ε = relative permittivity of fluid medium (dimensionless)
- κ = inverse screening length (1/distance units)
- ψ = surface potential (energy/charge units)

The last coefficient is optional. If not specified, the global yukawa/colloid cutoff is used.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.284.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs I,J and I != J, the A coefficient and cutoff distance for this pair style can be mixed. A is an energy value mixed like a LJ epsilon. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style supports the *pair_modify* shift option for the energy of the pair interaction.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.284.5 Restrictions

This style is part of the COLLOID package. It is only enabled if LAMMPS was built with that package. See the *Build package* page for more info.

This pair style requires that atoms be finite-size spheres with a diameter, as defined by the *atom_style sphere* command.

Per-particle polydispersity is not yet supported by this pair style; per-type polydispersity is allowed. This means all particles of the same type must have the same diameter. Each type can have a different diameter.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.284.6 Related commands

pair_coeff

4.284.7 Default

none

(Safran) Safran, Statistical Thermodynamics of Surfaces, Interfaces, And Membranes, Westview Press, ISBN: 978-0813340791 (2003).

4.285 pair_style zbl command

Accelerator Variants: *zbl/gpu*, *zbl/kk*, *zbl/omp*

4.285.1 Syntax

```
pair_style zbl inner outer
```

- inner = distance where switching function begins
- outer = global cutoff for ZBL interaction

4.285.2 Examples

```
pair_style zbl 3.0 4.0
pair_coeff * * 73.0 73.0
pair_coeff 1 1 14.0 14.0
```

4.285.3 Description

Style *zbl* computes the Ziegler-Biersack-Littmark (ZBL) screened nuclear repulsion for describing high-energy collisions between atoms. *(Ziegler)*. It includes an additional switching function that ramps the energy, force, and curvature smoothly to zero between an inner and outer cutoff. The potential energy due to a pair of atoms at a distance r_ij is given by:

$$E_{ij}^{ZBL} = \frac{1}{4\pi\varepsilon_0}\frac{Z_i Z_j\, e^2}{r_{ij}}\phi(r_{ij}/a) + S(r_{ij})$$

$$a = \frac{0.46850}{Z_i^{0.23} + Z_j^{0.23}}$$

$$\phi(x) = 0.18175e^{-3.19980x} + 0.50986e^{-0.94229x} + 0.28022e^{-0.40290x} + 0.02817e^{-0.20162x}$$

where e is the electron charge, ε_0 is the electrical permittivity of vacuum, and Z_i and Z_j are the nuclear charges of the two atoms. The switching function $S(r)$ is identical to that used by *pair_style lj/gromacs*. Here, the inner and outer cutoff are the same for all pairs of atom types.

The following coefficients must be defined for each pair of atom types via the *pair_coeff* command as in the examples above, or in the LAMMPS data file.

- Z_i (atomic number for first atom type, e.g. 13.0 for aluminum)
- Z_j (ditto for second atom type)

The values of Z_i and Z_j are normally equal to the atomic numbers of the two atom types. Thus, the user may optionally specify only the coefficients for each $i == i$ pair, and rely on the obvious mixing rule for cross interactions (see below). Note that when $i == i$ it is required that $Z_i == Z_j$. When used with *hybrid/overlay* and pairs are assigned to more than one sub-style, the mixing rule is not used and each pair of types interacting with the ZBL sub-style must be included in a pair_coeff command.

Note

The numerical values of the exponential decay constants in the screening function depend on the unit of distance. In the above equation they are given for units of Angstroms. LAMMPS will automatically convert these values to the distance unit of the specified LAMMPS *units* setting. The values of Z should always be given as multiples of a proton's charge, e.g. 29.0 for copper.

Styles with a *gpu*, *intel*, *kk*, *omp*, or *opt* suffix are functionally the same as the corresponding style without the suffix. They have been optimized to run faster, depending on your available hardware, as discussed on the *Accelerator packages* page. The accelerated styles take the same arguments and should produce the same results, except for round-off and precision issues.

These accelerated styles are part of the GPU, INTEL, KOKKOS, OPENMP, and OPT packages, respectively. They are only enabled if LAMMPS was built with those packages. See the *Build package* page for more info.

You can specify the accelerated styles explicitly in your input script by including their suffix, or you can use the *-suffix command-line switch* when you invoke LAMMPS, or you can use the *suffix* command in your input script.

See the *Accelerator packages* page for more instructions on how to use the accelerated styles effectively.

4.285.4 Mixing, shift, table, tail correction, restart, rRESPA info

For atom type pairs *i,j* and $i \neq i$, the Z_i and Z_j coefficients can be mixed by taking Z_i and Z_j from the values specified for $i == i$ and $j == j$ cases. When used with *hybrid/overlay* and pairs are assigned to more than one sub-style, the mixing rule is not used and each pair of types interacting with the ZBL sub-style must be included in a pair_coeff command. The *pair_modify* mix option has no effect on the mixing behavior

The ZBL pair style does not support the *pair_modify* shift option, since the ZBL interaction is already smoothed to 0.0 at the cutoff.

The *pair_modify* table option is not relevant for this pair style.

This pair style does not support the *pair_modify* tail option for adding long-range tail corrections to energy and pressure, since there are no corrections for a potential that goes to 0.0 at the cutoff.

This pair style does not write information to *binary restart files*, so pair_style and pair_coeff commands must be specified in an input script that reads a restart file.

This pair style can only be used via the *pair* keyword of the *run_style respa* command. It does not support the *inner*, *middle*, *outer* keywords.

4.285.5 Restrictions

none

4.285.6 Related commands

pair_coeff

4.285.7 Default

none

(Ziegler) J.F. Ziegler, J. P. Biersack and U. Littmark, "The Stopping and Range of Ions in Matter", Volume 1, Pergamon, 1985.

4.286 pair_style zero command

4.286.1 Syntax

```
pair_style zero cutoff [nocoeff] [full]
```

- zero = style name of this pair style
- cutoff = global cutoff (distance units)
- nocoeff = ignore all pair_coeff parameters (optional)
- full = build full neighbor list (optional)

4.286.2 Examples

```
pair_style zero 10.0
pair_style zero 5.0 nocoeff
pair_coeff * *
pair_coeff 1 2*4 3.0
```

4.286.3 Description

Define a global or per-type cutoff length for the purpose of building a neighbor list and acquiring ghost atoms, but do not compute any pairwise forces or energies.

This can be useful for fixes or computes which require a neighbor list to enumerate pairs of atoms within some cutoff distance, but when pairwise forces are not otherwise needed. Examples are the *fix bond/create*, *compute rdf*, *compute voronoi/atom* commands.

Note that the *comm_modify cutoff* command can be used to ensure communication of ghost atoms even when a pair style is not defined, but it will not trigger neighbor list generation.

The optional *nocoeff* flag allows to read data files with a PairCoeff section for any pair style. Similarly, any pair_coeff commands will only be checked for the atom type numbers and the rest ignored. In this case, only the global cutoff will be used.

Added in version 3Nov2022.

The optional *full* flag builds a full neighbor list instead of the default half neighbor list.

The following coefficients must be defined for each pair of atoms types via the *pair_coeff* command as in the examples above, or in the data file or restart files read by the *read_data* or *read_restart* commands, or by mixing as described below:

- cutoff (distance units)

This coefficient is optional. If not specified, the global cutoff specified in the pair_style command is used. If the pair_style has been specified with the optional *nocoeff* flag, then a cutoff pair coefficient is ignored.

4.286.4 Mixing, shift, table, tail correction, restart, rRESPA info

The cutoff distance for this pair style can be mixed. The default mix value is *geometric*. See the "pair_modify" command for details.

This pair style does not support the *pair_modify* shift, table, and tail options.

This pair style writes its information to *binary restart files*, so pair_style and pair_coeff commands do not need to be specified in an input script that reads a restart file.

This pair style supports the use of the *inner*, *middle*, and *outer* keywords of the *run_style respa* command.

4.286.5 Restrictions

none

4.286.6 Related commands

pair_style none

4.286.7 Default

none