CHAPTER

SEVEN

DIHEDRAL STYLES

7.1 dihedral_style charmm command

Accelerator Variants: *charmm/intel*, *charmm/kk*, *charmm/omp*

7.2 dihedral_style charmmfsw command

Accelerator Variants: *charmmfsw/kk*

7.2.1 Syntax

```
dihedral_style style
```

- style = *charmm* or *charmmfsw*

7.2.2 Examples

```
dihedral_style charmm
dihedral_style charmmfsw
dihedral_coeff  1 0.2 1 180 1.0
dihedral_coeff  2 1.8 1   0 1.0
dihedral_coeff  1 3.1 2 180 0.5
```

7.2.3 Description

The *charmm* and *charmmfsw* dihedral styles use the potential

$$E = K[1 + \cos(n\phi - d)]$$

See *(MacKerell)* for a description of the CHARMM force field. This dihedral style can also be used for the AMBER force field (see comment on weighting factors below). See *(Cornell)* for a description of the AMBER force field.

Note

The newer *charmmfsw* style was released in March 2017. We recommend it be used instead of the older *charmm* style when running a simulation with the CHARMM force field, either with long-range Coulombics or a Coulombic cutoff, via the *pair_style lj/charmmfsw/coul/long* and *pair_style lj/charmmfsw/coul/charmmfsh* commands respectively. Otherwise the older *charmm* style is fine to use. See the discussion below and more details on the *pair_style charmm* doc page.

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- *K* (energy)
- *n* (integer >= 0)
- *d* (integer value of degrees)
- weighting factor (1.0, 0.5, or 0.0)

The weighting factor is required to correct for double counting pairwise non-bonded Lennard-Jones interactions in cyclic systems or when using the CHARMM dihedral style with non-CHARMM force fields. With the CHARMM dihedral style, interactions between the first and fourth atoms in a dihedral are skipped during the normal non-bonded force computation and instead evaluated as part of the dihedral using special epsilon and sigma values specified with the *pair_coeff* command of pair styles that contain "lj/charmm" (e.g. *pair_style lj/charmm/coul/long*) In 6-membered rings, the same 1-4 interaction would be computed twice (once for the clockwise 1-4 pair in dihedral 1-2-3-4 and once in the counterclockwise dihedral 1-6-5-4) and thus the weighting factor has to be 0.5 in this case. In 4-membered or 5-membered rings, the 1-4 dihedral also is counted as a 1-2 or 1-3 interaction when going around the ring in the opposite direction and thus the weighting factor is 0.0, as the 1-2 and 1-3 exclusions take precedence.

Note that this dihedral weighting factor is unrelated to the scaling factor specified by the *special bonds* command which applies to all 1-4 interactions in the system. For CHARMM force fields, the special_bonds 1-4 interaction scaling factor should be set to 0.0. Since the corresponding 1-4 non-bonded interactions are computed with the dihedral. This means that if any of the weighting factors defined as dihedral coefficients (fourth coeff above) are non-zero, then you must use a pair style with "lj/charmm" and set the special_bonds 1-4 scaling factor to 0.0 (which is the default). Otherwise 1-4 non-bonded interactions in dihedrals will be computed twice.

For simulations using the CHARMM force field with a Coulombic cutoff, the difference between the *charmm* and *charmmfsw* styles is in the computation of the 1-4 non-bond interactions, though only if the distance between the two atoms is within the switching region of the pairwise potential defined by the corresponding CHARMM pair style, i.e. within the outer cutoff specified for the pair style. The *charmmfsw* style should only be used when using the corresponding *pair_style lj/charmmfsw/coul/charmmfsw* or *pair_style lj/charmmfsw/coul/long* commands. Use the *charmm* style with the older *pair_style* commands that have just "charmm" in their style name. See the discussion on the *CHARMM pair_style* page for details.

Note that for AMBER force fields, which use pair styles with "lj/cut", the special_bonds 1-4 scaling factor should be set to the AMBER defaults (1/2 and 5/6) and all the dihedral weighting factors (fourth coeff above) must be set to 0.0. In this case, you can use any pair style you wish, since the dihedral does not need any Lennard-Jones parameter information and will not compute any 1-4 non-bonded interactions. Likewise the *charmm* or *charmmfsw* styles are identical in this case since no 1-4 non-bonded interactions are computed.

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.

7.2.4 Restrictions

When using run_style *respa*, these dihedral styles must be assigned to the same r-RESPA level as *pair* or *outer*.

When used in combination with CHARMM pair styles, the 1-4 *special_bonds* scaling factors must be set to 0.0. Otherwise non-bonded contributions for these 1-4 pairs will be computed multiple times.

These dihedral styles can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

7.2.5 Related commands

dihedral_coeff , *pair_style lj/charmm variants*, *angle_style charmm*, *fix cmap*

7.2.6 Default

none

(Cornell) Cornell, Cieplak, Bayly, Gould, Merz, Ferguson, Spellmeyer, Fox, Caldwell, Kollman, JACS 117, 5179-5197 (1995).

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

7.3 dihedral_style class2 command

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

7.3.1 Syntax

```
dihedral_style class2
```

7.3.2 Examples

```
dihedral_style class2
dihedral_coeff 1 100 75 100 70 80 60
dihedral_coeff * mbt 3.5945 0.1704 -0.5490 1.5228
dihedral_coeff * ebt 0.3417 0.3264 -0.9036 0.1368 0.0 -0.8080 1.0119 1.1010
dihedral_coeff 2 at 0.0 -0.1850 -0.7963 -2.0220 0.0 -0.3991 110.2453 105.1270
dihedral_coeff * aat -13.5271 110.2453 105.1270
dihedral_coeff * bb13 0.0 1.0119 1.1010
```

7.3.3 Description

The *class2* dihedral style uses the potential

$$\begin{aligned}
E =& E_d + E_{mbt} + E_{ebt} + E_{at} + E_{aat} + E_{bb13} \\
E_d =& \sum_{n=1}^{3} K_n[1 - \cos(n\phi - \phi_n)] \\
E_{mbt} =& (r_{jk} - r_2)[A_1 \cos(\phi) + A_2 \cos(2\phi) + A_3 \cos(3\phi)] \\
E_{ebt} =& (r_{ij} - r_1)[B_1 \cos(\phi) + B_2 \cos(2\phi) + B_3 \cos(3\phi)] + \\
& (r_{kl} - r_3)[C_1 \cos(\phi) + C_2 \cos(2\phi) + C_3 \cos(3\phi)] \\
E_{at} =& (\theta_{ijk} - \theta_1)[D_1 \cos(\phi) + D_2 \cos(2\phi) + D_3 \cos(3\phi)] + \\
& (\theta_{jkl} - \theta_2)[E_1 \cos(\phi) + E_2 \cos(2\phi) + E_3 \cos(3\phi)] \\
E_{aat} =& M(\theta_{ijk} - \theta_1)(\theta_{jkl} - \theta_2) \cos(\phi) \\
E_{bb13} =& N(r_{ij} - r_1)(r_{kl} - r_3)
\end{aligned}$$

where E_d is the dihedral term, E_{mbt} is a middle-bond-torsion term, E_{ebt} is an end-bond-torsion term, E_{at} is an angle-torsion term, E_{aat} is an angle-angle-torsion term, and E_{bb13} is a bond-bond-13 term.

θ_1 and θ_2 are equilibrium angles and r_1, r_2, and r_3 are equilibrium bond lengths.

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

Coefficients for the E_d, E_{mbt}, E_{ebt}, E_{at}, E_{aat}, and E_{bb13} formulas must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands.

These are the 6 coefficients for the E_d formula:

- K_1 (energy)
- ϕ_1 (degrees)
- K_2 (energy)
- ϕ_2 (degrees)
- K_3 (energy)
- phi_3 (degrees)

For the E_{mbt} formula, each line in a *dihedral_coeff* command in the input script lists 5 coefficients, the first of which is *mbt* to indicate they are MiddleBondTorsion coefficients. In a data file, these coefficients should be listed under a *MiddleBondTorsion Coeffs* heading and you must leave out the *mbt*, i.e. only list 4 coefficients after the dihedral type.

- *mbt*
- A_1 (energy/distance)

- A_2 (energy/distance)
- A_3 (energy/distance)
- r_2 (distance)

For the E_{ebt} formula, each line in a *dihedral_coeff* command in the input script lists 9 coefficients, the first of which is *ebt* to indicate they are EndBondTorsion coefficients. In a data file, these coefficients should be listed under a *EndBondTorsion Coeffs* heading and you must leave out the *ebt*, i.e. only list 8 coefficients after the dihedral type.

- *ebt*
- B_1 (energy/distance)
- B_2 (energy/distance)
- B_3 (energy/distance)
- C_1 (energy/distance)
- C_2 (energy/distance)
- C_3 (energy/distance)
- r_1 (distance)
- r_3 (distance)

For the E_{at} formula, each line in a *dihedral_coeff* command in the input script lists 9 coefficients, the first of which is *at* to indicate they are AngleTorsion coefficients. In a data file, these coefficients should be listed under a *AngleTorsion Coeffs* heading and you must leave out the *at*, i.e. only list 8 coefficients after the dihedral type.

- *at*
- D_1 (energy)
- D_2 (energy)
- D_3 (energy)
- E_1 (energy)
- E_2 (energy)
- E_3 (energy)
- θ_1 (degrees)
- θ_2 (degrees)

θ_1 and θ_2 are specified in degrees, but LAMMPS converts them to radians internally; hence the various D and E are effectively energy per radian.

For the E_{aat} formula, each line in a *dihedral_coeff* command in the input script lists 4 coefficients, the first of which is *aat* to indicate they are AngleAngleTorsion coefficients. In a data file, these coefficients should be listed under a *AngleAngleTorsion Coeffs* heading and you must leave out the *aat*, i.e. only list 3 coefficients after the dihedral type.

- *aat*
- M (energy)
- θ_1 (degrees)
- θ_2 (degrees)

θ_1 and θ_2 are specified in degrees, but LAMMPS converts them to radians internally; hence M is effectively energy per radian^2.

For the E_{bb13} formula, each line in a *dihedral_coeff* command in the input script lists 4 coefficients, the first of which is *bb13* to indicate they are BondBond13 coefficients. In a data file, these coefficients should be listed under a *BondBond13 Coeffs* heading and you must leave out the *bb13*, i.e. only list 3 coefficients after the dihedral type.

- *bb13*
- *N* (energy/distance^2)
- r_1 (distance)
- r_3 (distance)

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.

7.3.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the CLASS2 package. See the *Build package* doc page for more info.

7.3.5 Related commands

dihedral_coeff

7.3.6 Default

none

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

7.4 dihedral_style cosine/shift/exp command

Accelerator Variants: *cosine/shift/exp/omp*

7.4.1 Syntax

```
dihedral_style cosine/shift/exp
```

7.4.2 Examples

```
dihedral_style cosine/shift/exp
dihedral_coeff 1 10.0 45.0 2.0
```

7.4.3 Description

The *cosine/shift/exp* dihedral style uses the potential

$$E = -U_{min}\frac{e^{-aU(\theta,\theta_0)} - 1}{e^a - 1} \quad \text{with} \quad U(\theta,\theta_0) = -0.5\left(1 + \cos(\theta - \theta_0)\right)$$

where U_{min}, θ, and a are defined for each dihedral type.

The potential is bounded between $[-U_{min} : 0]$ and the minimum is located at the angle θ_0. The a parameter can be both positive or negative and is used to control the spring constant at the equilibrium.

The spring constant is given by $k = ae^a \frac{U_{min}}{2(e^a-1)}$. For $a > 3$ and $\frac{k}{U_{min}} = \frac{a}{2}$ to better than 5% relative error. For negative values of the a parameter, the spring constant is essentially zero, and anharmonic terms takes over. The potential is furthermore well behaved in the limit $a \rightarrow 0$, where it has been implemented to linear order in a for $a < 0.001$.

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- U_{min} (energy)
- θ (angle)
- a (real number)

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.

7.4.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

7.4.5 Related commands

dihedral_coeff, *angle_style cosine/shift/exp*

7.4.6 Default

none

7.5 dihedral_style cosine/squared/restricted command

7.5.1 Syntax

```
dihedral_style cosine/squared/restricted
```

7.5.2 Examples

```
dihedral_style cosine/squared/restricted
dihedral_coeff 1 10.0 120
```

7.5.3 Description

Added in version 17Apr2024.

The *cosine/squared/restricted* dihedral style uses the potential

$$E = K[\cos(\phi) - \cos(\phi_0)]^2 / \sin^2(\phi)$$

, which is commonly used in the MARTINI force field.

See *(Bulacu)* for a description of the restricted dihedral for the MARTINI force field.

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- K (energy)
- ϕ_0 (degrees)

ϕ_0 is specified in degrees, but LAMMPS converts it to radians internally.

7.5.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.5.5 Related commands

dihedral_coeff

7.5.6 Default

none

(Bulacu) Bulacu, Goga, Zhao, Rossi, Monticelli, Periole, Tieleman, Marrink, J Chem Theory Comput, 9, 3282-3292 (2013).

7.6 dihedral_style fourier command

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

7.6.1 Syntax

```
dihedral_style fourier
```

7.6.2 Examples

```
dihedral_style fourier
dihedral_coeff 1 3 -0.846200 3 0.0 7.578800 1 0 0.138000 2 -180.0
```

7.6.3 Description

The *fourier* dihedral style uses the potential:

$$E = \sum_{i=1,m} K_i[1.0 + \cos(n_i\phi - d_i)]$$

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- m (integer >=1)
- K_1 (energy)
- n_1 (integer >= 0)
- d_1 (degrees)
- [. . .]

- K_m (energy)
- n_m (integer >= 0)
- d_m (degrees)

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.

7.6.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.6.5 Related commands

dihedral_coeff

7.6.6 Default

none

7.7 dihedral_style harmonic command

Accelerator Variants: *harmonic/intel*, *harmonic/kk*, *harmonic/omp*

7.7.1 Syntax

```
dihedral_style harmonic
```

7.7.2 Examples

```
dihedral_style harmonic
dihedral_coeff 1 80.0 1 2
```

7.7.3 Description

The *harmonic* dihedral style uses the potential

$$E = K[1 + d\cos(n\phi)]$$

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- K (energy)
- d (+1 or -1)
- n (integer >= 0)

Note

Here are important points to take note of when defining LAMMPS dihedral coefficients for the harmonic style, so that they are compatible with how harmonic dihedrals are defined by other force fields:

- The LAMMPS convention is that the trans position = 180 degrees, while in some force fields trans = 0 degrees.
- Some force fields reverse the sign convention on d.
- Some force fields let n be positive or negative which corresponds to $d = 1$ or $d = -1$ for the harmonic style.

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.

7.7.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

7.7.5 Related commands

dihedral_coeff

7.7.6 Default

none

7.8 dihedral_style helix command

Accelerator Variants: *helix/omp*

7.8.1 Syntax

```
dihedral_style helix
```

7.8.2 Examples

```
dihedral_style helix
dihedral_coeff 1 80.0 100.0 40.0
```

7.8.3 Description

The *helix* dihedral style uses the potential

$$E = A[1 - \cos(\theta)] + B[1 + \cos(3\theta)] + C[1 + \cos(\theta + \frac{\pi}{4})]$$

This coarse-grain dihedral potential is described in *(Guo)*. For dihedral angles in the helical region, the energy function is represented by a standard potential consisting of three minima, one corresponding to the trans (t) state and the other to gauche states (g+ and g-). The paper describes how the A, B and, C parameters are chosen so as to balance secondary (largely driven by local interactions) and tertiary structure (driven by long-range interactions).

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A (energy)
- B (energy)
- C (energy)

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.

7.8.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.8.5 Related commands

dihedral_coeff

7.8.6 Default

none

(Guo) Guo and Thirumalai, Journal of Molecular Biology, 263, 323-43 (1996).

7.9 dihedral_style hybrid command

Accelerator Variants: *hybrid/kk*

7.9.1 Syntax

```
dihedral_style hybrid style1 style2 ...
```

- style1,style2 = list of one or more dihedral styles

7.9.2 Examples

```
dihedral_style hybrid harmonic helix
dihedral_coeff 1 harmonic 6.0 1 3
dihedral_coeff 2* helix 10 10 10
```

7.9.3 Description

The *hybrid* style enables the use of multiple dihedral styles in one simulation. An dihedral style is assigned to each dihedral type. For example, dihedrals in a polymer flow (of dihedral type 1) could be computed with a *harmonic* potential and dihedrals in the wall boundary (of dihedral type 2) could be computed with a *helix* potential. The assignment of dihedral type to style is made via the *dihedral_coeff* command or in the data file.

In the dihedral_coeff commands, the name of a dihedral style must be added after the dihedral type, with the remaining coefficients being those appropriate to that style. In the example above, the 2 dihedral_coeff commands set dihedrals of dihedral type 1 to be computed with a *harmonic* potential with coefficients 6.0, 1, 3 for K, d, n. All other dihedral types (2-N) are computed with a *helix* potential with coefficients 10, 10, 10 for A, B, C.

If dihedral coefficients are specified in the data file read via the *read_data* command, then the same rule applies. E.g. "harmonic" or "helix", must be added after the dihedral type, for each line in the "Dihedral Coeffs" section, e.g.

```
Dihedral Coeffs

1 harmonic 6.0 1 3
2 helix 10 10 10
...
```

If *class2* is one of the dihedral hybrid styles, the same rule holds for specifying additional AngleTorsion (and EndBondTorsion, etc) coefficients either via the input script or in the data file. I.e. *class2* must be added to each line after the dihedral type. For lines in the AngleTorsion (or EndBondTorsion, etc) Coeffs section of the data file for dihedral types that are not *class2*, you must use an dihedral style of *skip* as a placeholder, e.g.

```
AngleTorsion Coeffs

1 skip
2 class2 1.0 1.0 1.0 3.0 3.0 3.0 30.0 50.0
...
```

Note that it is not necessary to use the dihedral style *skip* in the input script, since AngleTorsion (or EndBondTorsion, etc) coefficients need not be specified at all for dihedral types that are not *class2*.

A dihedral style of *none* with no additional coefficients can be used in place of a dihedral style, either in a input script dihedral_coeff command or in the data file, if you desire to turn off interactions for specific dihedral 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.

7.9.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

Unlike other dihedral styles, the hybrid dihedral style does not store dihedral coefficient info for individual sub-styles in *binary restart files* or *data files*. Thus when restarting a simulation, you need to re-specify the dihedral_coeff commands.

7.9.5 Related commands

dihedral_coeff

7.9.6 Default

none

7.10 dihedral_style lepton command

Accelerator Variants: *lepton/omp*

7.10.1 Syntax

```
dihedral_style lepton
```

7.10.2 Examples

```
dihedral_style lepton

dihedral_coeff  1 "k*(1 + d*cos(n*phi)); k=75.0; d=1; n=2"
dihedral_coeff  2 "45*(1-cos(4*phi))"
dihedral_coeff  2 "k2*cos(phi) + k3*cos(phi)^2; k2=100.0"
dihedral_coeff  3 "k*(phi-phi0)^2; k=85.0; phi0=120.0"
```

7.10.3 Description

Added in version 8Feb2023.

Dihedral style *lepton* computes dihedral interactions between four atoms forming a dihedral angle with a custom potential function. The potential function must be provided as an expression string using "phi" as the dihedral angle variable. For example *"200.0*(phi-120.0)^2"* represents a *quadratic dihedral* potential around a 120 degree dihedral angle with a force constant K of 200.0 energy units:

$$U_{dihedral,i} = K(\phi_i - \phi_0)^2$$

The Lepton library, that the *lepton* dihedral 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 "phi" and then uses that to compute the force between the dihedral atoms as defined by the topology data.

The potential function expression for each dihedral type is provided via the *dihedral_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 expression is in energy 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. Dihedral angles are internally computed in radians and thus the expression must assume "phi" is in radians.

7.10.4 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.

7.10.5 Restrictions

This dihedral style is part of the LEPTON package and only enabled if LAMMPS was built with this package. See the *Build package* page for more info.

7.10.6 Related commands

dihedral_coeff, *dihedral_style table*, *bond_style lepton*, *angle_style lepton*

7.10.7 Default

none

7.11 dihedral_style multi/harmonic command

Accelerator Variants: *multi/harmonic/omp*

7.11.1 Syntax

```
dihedral_style multi/harmonic
```

7.11.2 Examples

```
dihedral_style multi/harmonic
dihedral_coeff 1 20 20 20 20 20
```

7.11.3 Description

The *multi/harmonic* dihedral style uses the potential

$$E = \sum_{n=1,5} A_n \cos^{n-1}(\phi)$$

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- A_1 (energy)
- A_2 (energy)
- A_3 (energy)
- A_4 (energy)
- A_5 (energy)

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.

7.11.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

7.11.5 Related commands

dihedral_coeff

7.11.6 Default

none

7.12 dihedral_style nharmonic command

Accelerator Variants: *nharmonic/omp*

7.12.1 Syntax

```
dihedral_style nharmonic
```

7.12.2 Examples

```
dihedral_style nharmonic
dihedral_coeff * 3 10.0 20.0 30.0
```

7.12.3 Description

The *nharmonic* dihedral style uses the potential:

$$E = \sum_{i=1,n} A_i \cos^{i-1}(\phi)$$

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- n (integer >=1)
- A_1 (energy)
- A_2 (energy)
- ...
- A_n (energy)

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.

7.12.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.12.5 Related commands

dihedral_coeff

7.12.6 Default

none

7.13 dihedral_style none command

7.13.1 Syntax

```
dihedral_style none
```

7.13.2 Examples

```
dihedral_style none
```

7.13.3 Description

Using a dihedral style of none means dihedral forces and energies are not computed, even if quadruplets of dihedral atoms were listed in the data file read by the *read_data* command.

See the *dihedral_style zero* command for a way to calculate dihedral statistics, but compute no dihedral interactions.

7.13.4 Restrictions

none

7.13.5 Related commands

dihedral_style zero

7.13.6 Default

none

7.14 dihedral_style opls command

Accelerator Variants: *opls/intel*, *opls/kk*, *opls/omp*

7.14.1 Syntax

```
dihedral_style opls
```

7.14.2 Examples

```
dihedral_style opls
dihedral_coeff 1 1.740 -0.157 0.279 0.00   # CT-CT-CT-CT
dihedral_coeff 2 0.000 0.000 0.366 0.000   # CT-CT-CT-HC
dihedral_coeff 3 0.000 0.000 0.318 0.000   # HC-CT-CT-HC
```

7.14.3 Description

The *opls* dihedral style uses the potential

$$\begin{aligned} E = & \frac{1}{2}K_1[1+\cos(\phi)] + \frac{1}{2}K_2[1-\cos(2\phi)] + \\ & \frac{1}{2}K_3[1+\cos(3\phi)] + \frac{1}{2}K_4[1-\cos(4\phi)] \end{aligned}$$

Note that the usual 1/2 factor is not included in the K values.

This dihedral potential is used in the OPLS force field and is described in *(Watkins)*.

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- K_1 (energy)
- K_2 (energy)
- K_3 (energy)
- K_4 (energy)

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.

7.14.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the MOLECULE package. See the *Build package* doc page for more info.

7.14.5 Related commands

dihedral_coeff

7.14.6 Default

none

(Watkins) Watkins and Jorgensen, J Phys Chem A, 105, 4118-4125 (2001).

7.15 dihedral_style quadratic command

Accelerator Variants: *quadratic/omp*

7.15.1 Syntax

```
dihedral_style quadratic
```

7.15.2 Examples

```
dihedral_style quadratic
dihedral_coeff 100.0 80.0
```

7.15.3 Description

The *quadratic* dihedral style uses the potential:

$$E = K(\phi - \phi_0)^2$$

This dihedral potential can be used to keep a dihedral in a predefined value (cis=zero, right-hand convention is used).

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the data file or restart files read by the *read_data* or *read_restart* commands:

- K (energy)
- ϕ_0 (degrees)

ϕ_0 is specified in degrees, but LAMMPS converts it to radians internally; hence K is effectively energy per radian^2.

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.

7.15.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.15.5 Related commands

dihedral_coeff

7.15.6 Default

none

7.16 dihedral_style spherical command

7.16.1 Syntax

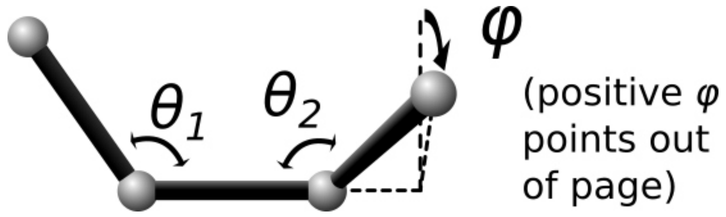
```
dihedral_style spherical
```

7.16.2 Examples

```
dihedral_coeff 1 1  286.1  1 124  1    1 90.0 0    1 90.0 0
dihedral_coeff 1 3  69.3   1 93.9 1    1 90   0    1 90   0  &
                    49.1   0 0.00 0    1 74.4 1    0 0.00 0  &
                    25.2   0 0.00 0    0 0.00 0    1 48.1 1
```

7.16.3 Description

The *spherical* dihedral style uses the potential:



$$E(\phi, \theta_1, \theta_2) = \sum_{i=1}^{N} C_i \, \Phi_i(\phi) \, \Theta_{1i}(\theta_1) \, \Theta_{2i}(\theta_2)$$
$$\Phi_i(\phi) = u_i - \cos((\phi - a_i)K_i)$$
$$\Theta_{1i}(\theta_1) = v_i - \cos((\theta_1 - b_i)L_i)$$
$$\Theta_{2i}(\theta_2) = w_i - \cos((\theta_2 - c_i)M_i)$$

For this dihedral style, the energy can be any function that combines the 4-body dihedral-angle (ϕ) and the two 3-body bond-angles (θ_1, θ_2). For this reason, there is usually no need to define 3-body "angle" forces separately for the atoms participating in these interactions. It is probably more efficient to incorporate 3-body angle forces into the dihedral interaction even if it requires adding additional terms to the expansion (as was done in the second example). A careful choice of parameters can prevent singularities that occur with traditional force-fields whenever theta1 or theta2 approach 0 or 180 degrees.

The last example above corresponds to an interaction with a single energy minima located near ϕ = 93.9, θ_1 = 74.4, θ_2 = 48.1 degrees, and it remains numerically stable at all angles (ϕ, θ_1, θ_2). In this example, the coefficients 49.1, and 25.2 can be physically interpreted as the harmonic spring constants for theta1 and theta2 around their minima. The coefficient 69.3 is the harmonic spring constant for phi after division by sin(74.4)*sin(48.1) (the minima positions for theta1 and theta2).

The following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above, or in the Dihedral Coeffs section of a data file read by the *read_data* command:

- n (integer >= 1)
- C_1 (energy)
- K_1 (typically an integer)
- a_1 (degrees)
- u_1 (typically 0.0 or 1.0)
- L_1 (typically an integer)
- b_1 (degrees, typically 0.0 or 90.0)
- v_1 (typically 0.0 or 1.0)
- M_1 (typically an integer)
- c_1 (degrees, typically 0.0 or 90.0)
- w_1 (typically 0.0 or 1.0)
- [. . .]
- C_n (energy)
- K_n (typically an integer)
- a_n (degrees)

- u_n (typically 0.0 or 1.0)
- L_n (typically an integer)
- b_n (degrees, typically 0.0 or 90.0)
- v_n (typically 0.0 or 1.0)
- M_n (typically an integer)
- c_n (degrees, typically 0.0 or 90.0)
- w_n (typically 0.0 or 1.0)

7.16.4 Restrictions

This dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.16.5 Related commands

dihedral_coeff

7.16.6 Default

none

7.17 dihedral_style table command

Accelerator Variants: *table/omp*

7.18 dihedral_style table/cut command

7.18.1 Syntax

```
dihedral_style style interpolation Ntable
```

- style = *table* or *table/cut*
- interpolation = *linear* or *spline* = method of interpolation
- Ntable = size of the internal lookup table

7.18.2 Examples

```
dihedral_style table spline 400
dihedral_style table linear 1000
dihedral_coeff 1 file.table DIH_TABLE1
dihedral_coeff 2 file.table DIH_TABLE2

dihedral_style table/cut spline 400
dihedral_style table/cut linear 1000
dihedral_coeff 1 aat 1.0 177 180 file.table DIH_TABLE1
dihedral_coeff 2 aat 0.5 170 180 file.table DIH_TABLE2
```

7.18.3 Description

The *table* and *table/cut* dihedral styles create interpolation tables of length *Ntable* from dihedral potential and derivative values listed in a file(s) as a function of the dihedral angle "phi". The files are read by the *dihedral_coeff* command. For dihedral style *table/cut* additionally an analytic cutoff that is quadratic in the bond-angle (theta) is applied in order to regularize the dihedral interaction.

The interpolation tables are created by fitting cubic splines to the file values and interpolating energy and derivative values at each of *Ntable* dihedral angles. During a simulation, these tables are used to interpolate energy and force values on individual atoms as needed. The interpolation is done in one of 2 styles: *linear* or *spline*.

For the *linear* style, the dihedral angle (phi) is used to find 2 surrounding table values from which an energy or its derivative is computed by linear interpolation.

For the *spline* style, cubic spline coefficients are computed and stored at each of the *Ntable* evenly-spaced values in the interpolated table. For a given dihedral angle (phi), the appropriate coefficients are chosen from this list, and a cubic polynomial is used to compute the energy and the derivative at this angle.

For dihedral style *table* the following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above.

- filename
- keyword

The filename specifies a file containing tabulated energy and derivative values. The keyword specifies which section of the file to read. The format of this file is the same for both dihedral styles and described below.

For dihedral style *table/cut* the following coefficients must be defined for each dihedral type via the *dihedral_coeff* command as in the example above.

- style (= aat)
- cutoff prefactor (unitless)
- cutoff angle1 (degrees)
- cutoff angle2 (degrees)
- filename
- keyword

The cutoff dihedral style uses a tabulated dihedral interaction with a cutoff function:

$$\begin{aligned} f(\theta) &= K & \theta &< \theta_1 \\ f(\theta) &= K\left(1 - \frac{(\theta-\theta_1)^2}{(\theta_2-\theta_1)^2}\right) & \theta_1 &< \theta < \theta_2 \end{aligned}$$

The cutoff includes a prefactor K to the cutoff function $f(\theta)$. While this value would ordinarily be 1, there may be situations where the value could be different.

The cutoff θ_1 specifies the angle (in degrees) below which the dihedral interaction is unmodified, i.e. the cutoff function is 1.

The cutoff function is applied between θ_1 and θ_2, which is the angle at which the cutoff function drops to zero. The value of zero effectively "turns off" the dihedral interaction.

The filename specifies a file containing tabulated energy and derivative values. The keyword specifies which section of the file to read. The format of this file is the same for both dihedral styles and described below.

Suitable tables for use with this dihedral style can be created using the Python code in the tools/tabulate folder of the LAMMPS source code distribution.

The format of a tabulated file is as follows (without the parenthesized comments). It can begin with one or more comment or blank lines.

```
# Table of the potential and its negative derivative

DIH_TABLE1                   (keyword is the first text on line)
N 30 DEGREES                 (N, NOF, DEGREES, RADIANS, CHECKU/F)
                      (blank line)
1 -168.0 -1.40351172223 0.0423346818422
2 -156.0 -1.70447981034 0.00811786522531
3 -144.0 -1.62956100432 -0.0184129719987
...
30 180.0 -0.707106781187 0.0719306095245

# Example 2: table of the potential. Forces omitted

DIH_TABLE2
N 30 NOF CHECKU testU.dat CHECKF testF.dat

1 -168.0 -1.40351172223
2 -156.0 -1.70447981034
3 -144.0 -1.62956100432
...
30 180.0 -0.707106781187
```

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 *dihedral_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.

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 the angle value, the third value is the energy (in energy units), and the fourth is -dE/d(phi) also in energy units). The third term is the energy of the 4-atom configuration for the specified angle. The fourth term (when present) is the negative derivative of the energy with respect to the angle (in degrees, or radians depending on whether the user selected DEGREES or RADIANS). Thus the units of the last term are still energy, not force. The dihedral angle values must increase from one line to the next.

Dihedral table splines are cyclic. There is no discontinuity at 180 degrees (or at any other angle). Although in the examples above, the angles range from -180 to 180 degrees, in general, the first angle in the list can have any value (positive, zero, or negative). However the *range* of angles represented in the table must be *strictly* less than 360 degrees

(2pi radians) to avoid angle overlap. (You may not supply entries in the table for both 180 and -180, for example.) If the user's table covers only a narrow range of dihedral angles, strange numerical behavior can occur in the large remaining gap.

Parameters:

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 *dihedral_style table* command. Let *Ntable* is the number of table entries requested dihedral_style command, and let *Nfile* be the parameter following "N" in the tabulated file ("30" in the sparse example above). 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 (which are evenly spaced over a 360 degree range, even if the angles in the file are not). The resulting tables of length *Ntable* are then used as described above, when computing energy and force for individual dihedral angles and their atoms. This means that if you want the interpolation tables of length *Ntable* to match exactly what is in the tabulated file (with effectively nopreliminary interpolation), you should set *Ntable* = *Nfile*. To ensure the nodal points in the user's file are aligned with the interpolated table entries, the angles in the table should be integer multiples of 360/*Ntable* degrees, or 2*PI/*Ntable* radians (depending on your choice of angle units).

The optional "NOF" keyword allows the user to omit the forces (negative energy derivatives) from the table file (normally located in the fourth column). In their place, forces will be calculated automatically by differentiating the potential energy function indicated by the third column of the table (using either linear or spline interpolation).

The optional "DEGREES" keyword allows the user to specify angles in degrees instead of radians (default).

The optional "RADIANS" keyword allows the user to specify angles in radians instead of degrees. (Note: This changes the way the forces are scaled in the fourth column of the data file.)

The optional "CHECKU" keyword is followed by a filename. This allows the user to save all of the *Ntable* different entries in the interpolated energy table to a file to make sure that the interpolated function agrees with the user's expectations. (Note: You can temporarily increase the *Ntable* parameter to a high value for this purpose. "*Ntable*" is explained above.)

The optional "CHECKF" keyword is analogous to the "CHECKU" keyword. It is followed by a filename, and it allows the user to check the interpolated force table. This option is available even if the user selected the "NOF" option.

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.

7.18.4 Restart, fix_modify, output, run start/stop, minimize info

These dihedral styles write the settings for the "dihedral_style table" or "dihedral_style table/cut" command to *binary restart files*, so a dihedral_style command does not need to specified in an input script that reads a restart file. However, the coefficient information loaded from the table file(s) is not stored in the restart file, since it is tabulated in the potential files. Thus, suitable dihedral_coeff commands do need to be specified in the restart input script after reading the restart file.

7.18.5 Restrictions

The *table* dihedral style can only be used if LAMMPS was built with the MOLECULE package. The *table/cut* dihedral style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* doc page for more info.

7.18.6 Related commands

dihedral_coeff

7.18.7 Default

none

7.19 dihedral_style zero command

7.19.1 Syntax

```
dihedral_style zero keyword
```

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

7.19.2 Examples

```
dihedral_style zero
dihedral_style zero nocoeff
dihedral_coeff *
```

7.19.3 Description

Using a dihedral style of zero means dihedral forces and energies are not computed, but the geometry of dihedral quadruplets is still accessible to other commands.

As an example, the *compute dihedral/local* command can be used to compute the theta values for the list of quadruplets of dihedral atoms listed in the data file read by the *read_data* command. If no dihedral style is defined, this command cannot be used.

The optional *nocoeff* flag allows to read data files with a DihedralCoeff section for any dihedral style. Similarly, any dihedral_coeff commands will only be checked for the dihedral type number and the rest ignored.

Note that the *dihedral_coeff* command must be used for all dihedral types, though no additional values are specified.

7.19.4 Restrictions

none

7.19.5 Related commands

none

dihedral_style none

7.19.6 Default

none