CHAPTER FIVE

BOND STYLES

5.1 bond_style bpm/rotational command

5.1.1 Syntax

```
bond_style bpm/rotational keyword value attribute1 attribute2 ...
```

- optional keyword = *overlay/pair* or *store/local* or *smooth* or *break*

```
store/local values = fix_ID N attributes ...
  * fix_ID = ID of associated internal fix to store data
  * N = prepare data for output every this many timesteps
  * attributes = zero or more of the below attributes may be appended

    id1, id2 = IDs of two atoms in the bond
    time = the timestep the bond broke
    x, y, z = the center of mass position of the two atoms when the bond broke (distance units)
    x/ref, y/ref, z/ref = the initial center of mass position of the two atoms (distance units)

overlay/pair value = yes or no
  bonded particles will still interact with pair forces

smooth value = yes or no
  smooths bond forces near the breaking point

normalize value = yes or no
  normalizes normal and shear forces by the reference length

break value = yes or no
  indicates whether bonds break during a run
```

5.1.2 Examples

```
bond_style bpm/rotational
bond_coeff 1 1.0 0.2 0.02 0.02 0.20 0.04 0.04 0.04 0.1 0.02 0.002 0.002

bond_style bpm/rotational store/local myfix 1000 time id1 id2
dump 1 all local 1000 dump.broken f_myfix[1] f_myfix[2] f_myfix[3]
dump_modify 1 write_header no
```

5.1.3 Description

Added in version 4May2022.

The *bpm/rotational* bond style computes forces and torques based on deviations from the initial reference state of the two atoms. The reference state is stored by each bond when it is first computed in the setup of a run. Data is then preserved across run commands and is written to *binary restart files* such that restarting the system will not reset the reference state of a bond.

Forces include a normal and tangential component. The base normal force has a magnitude of

$$f_r = k_r(r - r_0)$$

where k_r is a stiffness and r is the current distance and r_0 is the initial distance between the two particles.

A tangential force is applied perpendicular to the normal direction which is proportional to the tangential shear displacement with a stiffness of k_s. This tangential force also induces a torque. In addition, bending and twisting torques are also applied to particles which are proportional to angular bending and twisting displacements with stiffnesses of k_b and k_t, respectively. Details on the calculations of shear displacements and angular displacements can be found in *(Wang)* and *(Wang and Mora)*.

Bonds will break under sufficient stress. A breaking criterion is calculated

$$B = \max\left\{0, \frac{f_r}{f_{r,c}} + \frac{|f_s|}{f_{s,c}} + \frac{|\tau_b|}{\tau_{b,c}} + \frac{|\tau_t|}{\tau_{t,c}}\right\}$$

where $|f_s|$ is the magnitude of the shear force and $|\tau_b|$ and $|\tau_t|$ are the magnitudes of the bending and twisting torques, respectively. The corresponding variables $f_{r,c}$ $f_{s,c}$, $\tau_{b,c}$, and $\tau_{t,c}$ are critical limits to each force or torque. If B is ever equal to or exceeds one, the bond will break. This is done by setting the bond type to 0 such that forces and torques are no longer computed.

After computing the base magnitudes of the forces and torques, they can be optionally multiplied by an extra factor w to smoothly interpolate forces and torques to zero as the bond breaks. This term is calculated as $w = (1.0 - B^4)$. This smoothing factor can be added or removed by setting the *smooth* keyword to *yes* or *no*, respectively.

Finally, additional damping forces and torques are applied to the two particles. A force is applied proportional to the difference in the normal velocity of particles using a similar construction as dissipative particle dynamics *(Groot)*:

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

where γ_n is the damping strength, $\hat{r}$ is the radial normal vector, and $\vec{v}$ is the velocity difference between the two particles. Similarly, tangential forces are applied to each atom proportional to the relative differences in sliding velocities with a constant prefactor γ_s *(Wang et al.)* along with their associated torques. The rolling and twisting components of the relative angular velocities of the two atoms are also damped by applying torques with prefactors of γ_r and γ_t, respectively.

The following coefficients must be defined for each bond type via the *bond_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_r (force/distance units)
- k_s (force/distance units)
- k_t (force*distance/radians units)
- k_b (force*distance/radians units)
- $f_{r,c}$ (force units)
- $f_{s,c}$ (force units)
- $\tau_{b,c}$ (force*distance units)
- $\tau_{t,c}$ (force*distance units)
- γ_n (force/velocity units)
- γ_s (force/velocity units)
- γ_r (force*distance/velocity units)
- γ_t (force*distance/velocity units)

If the *normalize* keyword is set to *yes*, the radial and shear forces will be normalized by r_0 such that k_r and k_s must be given in force units.

By default, pair forces are not calculated between bonded particles. Pair forces can alternatively be overlaid on top of bond forces by setting the *overlay/pair* keyword to *yes*. These settings require specific *special_bonds* settings described in the restrictions. Further details can be found in the *how to* page on BPMs.

Added in version 28Mar2023.

If the *break* keyword is set to *no*, LAMMPS assumes bonds should not break during a simulation run. This will prevent some unnecessary calculation. However, if a bond reaches a damage criterion greater than one, it will trigger an error.

If the *store/local* keyword is used, an internal fix will track bonds that break during the simulation. Whenever a bond breaks, data is processed and transferred to an internal fix labeled *fix_ID*. This allows the local data to be accessed by other LAMMPS commands. Following this optional keyword, a list of one or more attributes is specified. These include the IDs of the two atoms in the bond. The other attributes for the two atoms include the timestep during which the bond broke and the current/initial center of mass position of the two atoms.

Data is continuously accumulated 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.

Note that when unbroken bonds are dumped to a file via the *dump local* command, bonds with type 0 (broken bonds) are not included. The *delete_bonds* command can also be used to query the status of broken bonds or permanently delete them, e.g.:

```
delete_bonds all stats
delete_bonds all bond 0 remove
```

5.1.4 Restart and other info

This bond style writes the reference state of each bond to *binary restart files*. Loading a restart file will properly resume bonds. However, the reference state is NOT written to data files. Therefore reading a data file will not restore bonds and will cause their reference states to be redefined.

If the *store/local* option is used, an internal fix will calculate 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, broken bonds. 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.

The single() function of this bond style returns 0.0 for the energy of a bonded interaction, since energy is not conserved in these dissipative potentials. It also returns only the normal component of the bonded interaction force. However, the single() function also calculates 7 extra bond quantities. The first 4 are data from the reference state of the bond including the initial distance between particles r_0 followed by the x, y, and z components of the initial unit vector pointing to particle I from particle J. The next 3 quantities (5-7) are the x, y, and z components of the total force, including normal and tangential contributions, acting on particle I.

These extra quantities can be accessed by the *compute bond/local* command, as *b1*, *b2*, . . . , *b7*.

5.1.5 Restrictions

This bond 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.

By default if pair interactions between bonded atoms are to be disabled, this bond style requires setting

```
special_bonds lj 0 1 1 coul 1 1 1
```

and *newton* must be set to bond off. If the *overlay/pair* keyword is set to *yes*, this bond style alternatively requires setting

```
special_bonds lj/coul 1 1 1
```

The *bpm/rotational* style requires *atom style bpm/sphere*.

5.1.6 Related commands

bond_coeff, *fix nve/bpm/sphere*

5.1.7 Default

The option defaults are *overlay/pair* = *no*, *smooth* = *yes*, *normalize* = *no*, and *break* = *yes*

(Wang) Wang, Acta Geotechnica, 4, p 117-127 (2009).

(Wang and Mora) Wang, Mora, Advances in Geocomputing, 119, p 183-228 (2009).

(Groot) Groot and Warren, J Chem Phys, 107, 4423-35 (1997).

(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.

5.2 bond_style bpm/spring command

5.2.1 Syntax

```
bond_style bpm/spring keyword value attribute1 attribute2 ...
```

- optional keyword = *overlay/pair* or *store/local* or *smooth* or *break*

```
store/local values = fix_ID N attributes ...
  * fix_ID = ID of associated internal fix to store data
  * N = prepare data for output every this many timesteps
  * attributes = zero or more of the below attributes may be appended

    id1, id2 = IDs of two atoms in the bond
    time = the timestep the bond broke
    x, y, z = the center of mass position of the two atoms when the bond broke (distance units)
    x/ref, y/ref, z/ref = the initial center of mass position of the two atoms (distance units)

overlay/pair value = yes or no
  bonded particles will still interact with pair forces

smooth value = yes or no
  smooths bond forces near the breaking point

normalize value = yes or no
  normalizes bond forces by the reference length

break value = yes or no
  indicates whether bonds break during a run
```

5.2.2 Examples

```
bond_style bpm/spring
bond_coeff 1 1.0 0.05 0.1

bond_style bpm/spring myfix 1000 time id1 id2
dump 1 all local 1000 dump.broken f_myfix[1] f_myfix[2] f_myfix[3]
dump_modify 1 write_header no
```

5.2.3 Description

Added in version 4May2022.

The *bpm/spring* bond style computes forces based on deviations from the initial reference state of the two atoms. The reference state is stored by each bond when it is first computed in the setup of a run. Data is then preserved across run commands and is written to *binary restart files* such that restarting the system will not reset the reference state of a bond.

This bond style only applies central-body forces which conserve the translational and rotational degrees of freedom of a bonded set of particles based on a model described by Clemmer and Robbins *(Clemmer)*. The force has a magnitude of

$$F = k(r - r_0)w$$

where k is a stiffness, r is the current distance and r_0 is the initial distance between the two particles, and w is an optional smoothing factor discussed below. Bonds will break at a strain of ϵ_c. This is done by setting the bond type to 0 such that forces are no longer computed.

An additional damping force is applied to the bonded particles. This forces is proportional to the difference in the normal velocity of particles using a similar construction as dissipative particle dynamics *(Groot)*:

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

where γ is the damping strength, $\hat{r}$ is the radial normal vector, and $\vec{v}$ is the velocity difference between the two particles.

The smoothing factor w can be added or removed by setting the *smooth* keyword to *yes* or *no*, respectively. It is constructed such that forces smoothly go to zero, avoiding discontinuities, as bonds approach the critical strain

$$w = 1.0 - \left(\frac{r - r_0}{r_0 \epsilon_c} \right)^8 .$$

The following coefficients must be defined for each bond type via the *bond_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 (force/distance units)
- ϵ_c (unit less)
- γ (force/velocity units)

If the *normalize* keyword is set to *yes*, the elastic bond force will be normalized by r_0 such that k must be given in force units.

By default, pair forces are not calculated between bonded particles. Pair forces can alternatively be overlaid on top of bond forces by setting the *overlay/pair* keyword to *yes*. These settings require specific *special_bonds* settings described in the restrictions. Further details can be found in the *how to* page on BPMs.

Added in version 28Mar2023.

If the *break* keyword is set to *no*, LAMMPS assumes bonds should not break during a simulation run. This will prevent some unnecessary calculation. However, if a bond reaches a strain greater than ϵ_c, it will trigger an error.

If the *store/local* keyword is used, an internal fix will track bonds that break during the simulation. Whenever a bond breaks, data is processed and transferred to an internal fix labeled *fix_ID*. This allows the local data to be accessed by other LAMMPS commands. Following this optional keyword, a list of one or more attributes is specified. These include the IDs of the two atoms in the bond. The other attributes for the two atoms include the timestep during which the bond broke and the current/initial center of mass position of the two atoms.

Data is continuously accumulated 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.

Note that when unbroken bonds are dumped to a file via the *dump local* command, bonds with type 0 (broken bonds) are not included. The *delete_bonds* command can also be used to query the status of broken bonds or permanently delete them, e.g.:

```
delete_bonds all stats
delete_bonds all bond 0 remove
```

5.2.4 Restart and other info

This bond style writes the reference state of each bond to *binary restart files*. Loading a restart file will properly restore bonds. However, the reference state is NOT written to data files. Therefore reading a data file will not restore bonds and will cause their reference states to be redefined.

If the *store/local* option is used, an internal fix will calculate 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, broken bonds. 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.

The single() function of this bond style returns 0.0 for the energy of a bonded interaction, since energy is not conserved in these dissipative potentials. The single() function also calculates an extra bond quantity, the initial distance r_0. This extra quantity can be accessed by the *compute bond/local* command as *b1*.

5.2.5 Restrictions

This bond 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.

By default if pair interactions between bonded atoms are to be disabled, this bond style requires setting

```
special_bonds lj 0 1 1 coul 1 1 1
```

and *newton* must be set to bond off. If the *overlay/pair* keyword is set to *yes*, this bond style alternatively requires setting

```
special_bonds lj/coul 1 1 1
```

5.2.6 Related commands

bond_coeff, *pair bpm/spring*

5.2.7 Default

The option defaults are *overlay/pair* = *no*, *smooth* = *yes*, *normalize* = *no*, and *break* = *yes*

(Clemmer) Clemmer and Robbins, Phys. Rev. Lett. (2022).

(Groot) Groot and Warren, J Chem Phys, 107, 4423-35 (1997).

5.3 bond_style class2 command

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

5.3.1 Syntax

```
bond_style class2
```

5.3.2 Examples

```
bond_style class2
bond_coeff 1 1.0 100.0 80.0 80.0
```

5.3.3 Description

The *class2* bond style uses the potential

$$E = K_2(r - r_0)^2 + K_3(r - r_0)^3 + K_4(r - r_0)^4$$

where r_0 is the equilibrium bond distance.

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

The following coefficients must be defined for each bond type via the *bond_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_0 (distance)
- K_2 (energy/distance^2)
- K_3 (energy/distance^3)
- K_4 (energy/distance^4)

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.

5.3.4 Restrictions

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

5.3.5 Related commands

bond_coeff, *delete_bonds*

5.3.6 Default

none

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

5.4 bond_style fene command

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

5.5 bond_style fene/nm command

5.5.1 Syntax

```
bond_style fene
bond_style fene/nm
```

5.5.2 Examples

```
bond_style fene
bond_coeff 1 30.0 1.5 1.0 1.0

bond_style fene/nm
bond_coeff 1 2.25344 1.5 1.0 1.12246 2 6
```

5.5.3 Description

The *fene* bond style uses the potential

$$E = -0.5KR_0^2 \ln\left[1-\left(\frac{r}{R_0}\right)^2\right] + 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^6\right]+\epsilon$$

to define a finite extensible nonlinear elastic (FENE) potential *(Kremer)*, used for bead-spring polymer models. The first term is attractive, the second Lennard-Jones term is repulsive. The first term extends to R_0, the maximum extent of the bond. The second term is cutoff at $2^{\frac{1}{6}}\sigma$, the minimum of the LJ potential.

The *fene/nm* bond style substitutes the standard LJ potential with the generalized LJ potential in the same form as in pair style *nm/cut*. The bond energy is then given by

$$E = -0.5KR_0^2 \ln\left[1 - \left(\frac{r}{R_0}\right)^2\right] + \frac{E_0}{(n-m)}\left[m\left(\frac{r_0}{r}\right)^n - n\left(\frac{r_0}{r}\right)^m\right]$$

Similar to the *fene* style, the generalized Lennard-Jones is cut off at the potential minimum, r_0, to be repulsive only. The following coefficients must be defined for each bond type via the *bond_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/distance^2)
- R_0 (distance)
- ϵ (energy)
- σ (distance)

For the *fene/nm* style, the following coefficients are used. Please note, that the standard LJ potential and thus the regular FENE potential is recovered for (n=12 m=6) and $r_0 = 2^{\frac{1}{6}}\sigma$.

- K (energy/distance^2)
- R_0 (distance)
- E_0 (energy)
- r_0 (distance)
- n (unitless)
- m (unitless)

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.

5.5.4 Restrictions

The *fene* bond style can only be used if LAMMPS was built with the MOLECULE package; the *fene/nm* bond style can only be used if LAMMPS was built with the EXTRA-MOLECULE package. See the *Build package* page for more info.

You typically should specify *special_bonds fene* or *special_bonds lj/coul 0 1 1* to use this bond style. LAMMPS will issue a warning it that's not the case.

5.5.5 Related commands

bond_coeff, *delete_bonds*, *pair style lj/cut*, *pair style nm/cut*.

5.5.6 Default

none

(Kremer) Kremer, Grest, J Chem Phys, 92, 5057 (1990).

5.6 bond_style fene/expand command

Accelerator Variants: *fene/expand/omp*

5.6.1 Syntax

```
bond_style fene/expand
```

5.6.2 Examples

```
bond_style fene/expand
bond_coeff 1 30.0 1.5 1.0 1.0 0.5
```

5.6.3 Description

The *fene/expand* bond style uses the potential

$$E = -0.5KR_0^2 \ln\left[1 - \left(\frac{(r-\Delta)}{R_0}\right)^2\right] + 4\epsilon\left[\left(\frac{\sigma}{(r-\Delta)}\right)^{12} - \left(\frac{\sigma}{(r-\Delta)}\right)^6\right] + \epsilon$$

to define a finite extensible nonlinear elastic (FENE) potential *(Kremer)*, used for bead-spring polymer models. The first term is attractive, the second Lennard-Jones term is repulsive.

The *fene/expand* bond style is similar to *fene* except that an extra shift factor of Δ (positive or negative) is added to r to effectively change the bead size of the bonded atoms. The first term now extends to $R_0 + \Delta$ and the second term is cutoff at $2^{\frac{1}{6}}\sigma + \Delta$.

The following coefficients must be defined for each bond type via the *bond_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/distance^2)
- R_0 (distance)
- ϵ (energy)
- σ (distance)
- Δ (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.

5.6.4 Restrictions

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

You typically should specify *special_bonds fene* or *special_bonds lj/coul 0 1 1* to use this bond style. LAMMPS will issue a warning it that's not the case.

5.6.5 Related commands

bond_coeff, *delete_bonds*

5.6.6 Default

none

(Kremer) Kremer, Grest, J Chem Phys, 92, 5057 (1990).

5.7 bond_style gaussian command

5.7.1 Syntax

```
bond_style gaussian
```

5.7.2 Examples

```
bond_style gaussian
bond_coeff 1 300.0 2 0.0128 0.375 3.37 0.0730 0.148 3.63
```

5.7.3 Description

The *gaussian* bond style uses the potential:

$$E = -k_B T ln \left(\sum_{i=1}^{n} \frac{A_i}{w_i \sqrt{\pi/2}} exp \left(\frac{-2(r-r_i)^2}{w_i^2} \right) \right)$$

This analytical form is a suitable potential for obtaining mesoscale effective force fields which can reproduce target atomistic distributions *(Milano)*

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

- T temperature at which the potential was derived
- n (integer >=1)
- A_1 (> 0, distance)
- w_1 (> 0, distance)
- r_1 (>= 0, distance)
- $\dots$
- A_n (> 0, distance)
- w_n (> 0, distance)
- r_n (>= 0, distance)

5.7.4 Restrictions

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

5.7.5 Related commands

bond_coeff

5.7.6 Default

none

(Milano) G. Milano, S. Goudeau, F. Mueller-Plathe, J. Polym. Sci. B Polym. Phys. 43, 871 (2005).

5.8 bond_style gromos command

Accelerator Variants: *gromos/omp*

5.8.1 Syntax

```
bond_style gromos
```

5.8.2 Examples

```
bond_style gromos
bond_coeff 5 80.0 1.2
```

5.8.3 Description

The *gromos* bond style uses the potential

$$E = K(r^2 - r_0^2)^2$$

where r_0 is the equilibrium bond distance. Note that the usual 1/4 factor is included in K.

The following coefficients must be defined for each bond type via the *bond_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/distance^4)
- r_0 (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.

5.8.4 Restrictions

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

5.8.5 Related commands

bond_coeff, *delete_bonds*

5.8.6 Default

none

5.9 bond_style harmonic command

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

5.9.1 Syntax

```
bond_style harmonic
```

5.9.2 Examples

```
bond_style harmonic
bond_coeff 5 80.0 1.2
```

5.9.3 Description

The *harmonic* bond style uses the potential

$$E = K(r - r_0)^2$$

where r_0 is the equilibrium bond distance. Note that the usual 1/2 factor is included in K.

The following coefficients must be defined for each bond type via the *bond_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/distance^2)
- r_0 (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.

5.9.4 Restrictions

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

5.9.5 Related commands

bond_coeff, *delete_bonds*

5.9.6 Default

none

5.10 bond_style harmonic/restrain command

5.10.1 Syntax

```
bond_style harmonic/restrain
```

5.10.2 Examples

```
bond_style harmonic
bond_coeff 5 80.0
```

5.10.3 Description

Added in version 28Mar2023.

The *harmonic/restrain* bond style uses the potential

$$E = K(r - r_{t=0})^2$$

where $r_{t=0}$ is the distance between the bonded atoms at the beginning of the first *run* or *minimize* command after the bond style has been defined (*t=0*). Note that the usual 1/2 factor is included in K. This will effectively restrain bonds to their initial length, whatever that is. This is where this bond style differs from *bond style harmonic* where the bond length is set through the per bond type coefficients.

The following coefficient must be defined for each bond type via the *bond_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/distance^2)

This bond style differs from other options to add harmonic restraints like *fix restrain* or *pair style list* or *fix colvars* in that it requires a bond topology, and thus the defined bonds will trigger exclusion of special neighbors from the neighbor list according to the *special_bonds* settings.

5.10.4 Restart info

This bond style supports the *write_restart* and *read_restart* commands. The state of the initial bond lengths is stored with restart files and read back.

5.10.5 Restrictions

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

This bond style maintains internal data to determine the original bond lengths $r_{t=0}$. This information will be written to *binary restart files* but **not** to *data files*. Thus, continuing a simulation is *only* possible with *read_restart*. When using the *read_data command*, the reference bond lengths $r_{t=0}$ will be re-initialized from the current geometry.

This bond style cannot be used with *fix shake or fix rattle*, with *fix filter/corotate*, or any *tip4p pair style* since there is no specific equilibrium distance for a given bond type.

5.10.6 Related commands

bond_coeff, *bond_harmonic*, *fix restrain*, *pair style list*

5.10.7 Default

none

5.11 bond_style harmonic/shift command

Accelerator Variants: *harmonic/shift/omp*

5.11.1 Syntax

```
bond_style harmonic/shift
```

5.11.2 Examples

```
bond_style harmonic/shift
bond_coeff 5 10.0 0.5 1.0
```

5.11.3 Description

The *harmonic/shift* bond style is a shifted harmonic bond that uses the potential

$$E = \frac{U_{\text{min}}}{(r_0 - r_c)^2}\left[(r - r_0)^2 - (r_c - r_0)^2\right]$$

where r_0 is the equilibrium bond distance, and r_c the critical distance. The potential is $-U_{\text{min}}$ at $r0$ and zero at r_c. The spring constant is $k = U_{\text{min}}/[2(r_0 - r_c)^2]$.

The following coefficients must be defined for each bond type via the *bond_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)
- r_0 (distance)
- r_c (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.

5.11.4 Restrictions

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

5.11.5 Related commands

bond_coeff, *delete_bonds*, *bond_harmonic*

5.11.6 Default

none

5.12 bond_style harmonic/shift/cut command

Accelerator Variants: *harmonic/shift/cut/omp*

5.12.1 Syntax

```
bond_style harmonic/shift/cut
```

5.12.2 Examples

```
bond_style harmonic/shift/cut
bond_coeff 5 10.0 0.5 1.0
```

5.12.3 Description

The *harmonic/shift/cut* bond style is a shifted harmonic bond that uses the potential

$$E = \frac{U_{\min}}{(r_0 - r_c)^2} \left[(r - r_0)^2 - (r_c - r_0)^2 \right]$$

where r_0 is the equilibrium bond distance, and rc the critical distance. The bond potential is zero for distances $r > r_c$. The potential is $-U_{\min}$ at r_0 and zero at r_c. The spring constant is $k = U_{\min}/[2(r_0 - r_c)^2]$.

The following coefficients must be defined for each bond type via the *bond_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)
- r_0 (distance)
- r_c (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.

5.12.4 Restrictions

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

5.12.5 Related commands

bond_coeff, *delete_bonds*, *bond_harmonic*, *bond_style harmonic/shift*

5.12.6 Default

none

5.13 bond_style hybrid command

Accelerator Variants: *hybrid/kk*

5.13.1 Syntax

```
bond_style hybrid style1 style2 ...
```

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

5.13.2 Examples

```
bond_style hybrid harmonic fene
bond_coeff 1 harmonic 80.0 1.2
bond_coeff 2* fene 30.0 1.5 1.0 1.0
```

5.13.3 Description

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

In the bond_coeff commands, the name of a bond style must be added after the bond type, with the remaining coefficients being those appropriate to that style. In the example above, the 2 bond_coeff commands set bonds of bond type 1 to be computed with a *harmonic* potential with coefficients 80.0, 1.2 for K, r_0. All other bond types (2-N) are computed with a *fene* potential with coefficients 30.0, 1.5, 1.0, 1.0 for K, R_0, ϵ, σ.

If bond coefficients are specified in the data file read via the *read_data* command, then the same rule applies. E.g. “harmonic” or “fene” must be added after the bond type, for each line in the “Bond Coeffs” section, e.g.

```
Bond Coeffs

1 harmonic 80.0 1.2
2 fene 30.0 1.5 1.0 1.0
...
```

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

5.13.4 Restrictions

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

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

5.13.5 Related commands

bond_coeff, *delete_bonds*

5.13.6 Default

none

5.14 bond_style lepton command

Accelerator Variants: *lepton/omp*

5.14.1 Syntax

```
bond_style style args
```

- style = *lepton*
- args = optional arguments

```
args = auto_offset or no_offset
  auto_offset = offset the potential energy so that the value at r0 is 0.0 (default)
  no_offset = do not offset the potential energy
```

5.14.2 Examples

```
bond_style lepton
bond_style lepton no_offset

bond_coeff  1  1.5 "k*r^2; k=250.0"
bond_coeff  2  1.1 "k2*r^2 + k3*r^3 + k4*r^4; k2=300.0; k3=-100.0; k4=50.0"
bond_coeff  3  1.3 "k*r^2; k=350.0"
```

5.14.3 Description

Added in version 8Feb2023.

Bond style *lepton* computes bonded interactions between two atoms with a custom function. The potential function must be provided as an expression string using "r" as the distance variable relative to the reference distance r_0 which is provided as a bond coefficient. For example *"200.0*r^2"* represents a harmonic potential with a force constant K of 200.0 energy units:

$$U_{bond,i} = K(r_i - r_0)^2 = Kr^2 \qquad r = r_i - r_0$$

Changed in version 7Feb2024.

By default the potential energy U is shifted so that he value U is 0.0 for $r = r_0$. This is equivalent to using the optional keyword *auto_offset*. When using the keyword *no_offset* instead, the potential energy is not shifted.

The Lepton library, that the *lepton* bond 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 atom pairs forming bonds as defined by the topology data.

The following coefficients must be defined for each bond type via the *bond_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)
- r_0 (distance)

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 r_0 is the "equilibrium distance". The potential energy function in the Lepton expression is shifted in such a way, that the potential energy is 0 for a bond length $r_i == r_0$.

5.14.4 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.

5.14.5 Restrictions

This bond 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.

5.14.6 Related commands

bond_coeff, *bond_style table*, *bond_write*, *angle_style lepton*, *dihedral_style lepton*

5.14.7 Default

none

5.15 bond_style mesocnt command

5.15.1 Syntax

```
bond_style mesocnt
```

5.15.2 Examples

```
bond_style mesocnt
bond_coeff 1 C 10 10 20.0
bond_coeff 4 custom 800.0 10.0
```

5.15.3 Description

Added in version 15Sep2022.

The *mesocnt* bond style is a wrapper for the *harmonic* style, and uses the potential

$$E = K(r - r_0)^2$$

where r_0 is the equilibrium bond distance. Note that the usual 1/2 factor is included in K. The style implements parameterization presets of K for mesoscopic simulations of carbon nanotubes based on the atomistic simulations of *(Srivastava)*.

Other presets can be readily implemented in the future.

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

- preset = *C* or *custom*
- additional parameters depending on preset

Preset *C* is for carbon nanotubes, and the additional parameters are:

- chiral index *n* (unitless)
- chiral index *m* (unitless)
- r_0 (distance)

Preset *custom* is simply a direct wrapper for the *harmonic* style, and the additional parameters are:

- K (energy/distance^2)
- r_0 (distance)

5.15.4 Restrictions

This bond style can only be used if LAMMPS was built with the MOLECULE and MESONT packages. See the *Build package* page for more info.

5.15.5 Related commands

bond_coeff, *delete_bonds*

5.15.6 Default

none

(Srivastava) Zhigilei, Wei and Srivastava, Phys. Rev. B 71, 165417 (2005).

5.16 bond_style mm3 command

5.16.1 Syntax

```
bond_style mm3
```

5.16.2 Examples

```
bond_style mm3
bond_coeff 1 100.0 107.0
```

5.16.3 Description

The *mm3* bond style uses the potential that is anharmonic in the bond as defined in *(Allinger)*

$$E = K(r - r_0)^2 \left[1 - 2.55(r - r_0) + \frac{7}{12} 2.55^2 (r - r_0)^2\right]$$

where r_0 is the equilibrium value of the bond, and K is a prefactor. The anharmonic prefactors have units $\mathring{A}^{-n}$: $-2.55\mathring{A}^{-1}$ and $\frac{7}{12}2.55^2\mathring{A}^{-2}$. The code takes care of the necessary unit conversion for these factors internally. Note that the MM3 papers contain an error in Eq (1): $\frac{7}{12}2.55$ should be replaced with $\frac{7}{12}2.55^2$

The following coefficients must be defined for each bond type via the *bond_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/distance^2)
- r_0 (distance)

5.16.4 Restrictions

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

5.16.5 Related commands

bond_coeff

5.16.6 Default

none

(Allinger) Allinger, Yuh, Lii, JACS, 111(23), 8551-8566 (1989),

5.17 bond_style morse command

Accelerator Variants: *morse/omp*

5.17.1 Syntax

```
bond_style morse
```

5.17.2 Examples

```
bond_style morse
bond_coeff 5 1.0 2.0 1.2
```

5.17.3 Description

The *morse* bond style uses the potential

$$E = D\left[1 - e^{-\alpha(r-r_0)}\right]^2$$

where r_0 is the equilibrium bond distance, α is a stiffness parameter, and D determines the depth of the potential well.

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

- D (energy)
- α (inverse distance)
- r_0 (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.

5.17.4 Restrictions

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

5.17.5 Related commands

bond_coeff, *delete_bonds*

5.17.6 Default

none

5.18 bond_style none command

5.18.1 Syntax

```
bond_style none
```

5.18.2 Examples

```
bond_style none
```

5.18.3 Description

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

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

5.18.4 Restrictions

none

5.18.5 Related commands

none

bond_style zero

5.18.6 Default

none

5.19 bond_style nonlinear command

Accelerator Variants: *nonlinear/omp*

5.19.1 Syntax

```
bond_style nonlinear
```

5.19.2 Examples

```
bond_style nonlinear
bond_coeff 2 100.0 1.1 1.4
```

5.19.3 Description

The *nonlinear* bond style uses the potential

$$E = \frac{\epsilon (r - r_0)^2}{[\lambda^2 - (r - r_0)^2]}$$

to define an anharmonic spring *(Rector)* of equilibrium length r_0 and maximum extension lamda.

The following coefficients must be defined for each bond type via the *bond_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)
- r_0 (distance)
- λ (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.

5.19.4 Restrictions

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

5.19.5 Related commands

bond_coeff, *delete_bonds*

5.19.6 Default

none

(Rector) Rector, Van Swol, Henderson, Molecular Physics, 82, 1009 (1994).

5.20 bond_style oxdna/fene command

5.21 bond_style oxdna2/fene command

5.22 bond_style oxrna2/fene command

5.22.1 Syntax

```
bond_style oxdna/fene

bond_style oxdna2/fene

bond_style oxrna2/fene
```

5.22.2 Examples

```
# LJ units
bond_style oxdna/fene
bond_coeff * 2.0 0.25 0.7525

bond_style oxdna2/fene
bond_coeff * 2.0 0.25 0.7564

bond_style oxrna2/fene
bond_coeff * 2.0 0.25 0.76107

bond_style oxdna/fene
bond_coeff * oxdna_lj.cgdna

# Real units
```

(continues on next page)

(continued from previous page)

```
bond_style oxdna/fene
bond_coeff * 11.92337812042065 2.1295 6.409795

bond_style oxdna2/fene
bond_coeff * 11.92337812042065 2.1295 6.4430152

bond_style oxrna2/fene
bond_coeff * 11.92337812042065 2.1295 6.482800913

bond_style oxrna2/fene
bond_coeff * oxrna2_real.cgdna
```

Note

The coefficients in the above examples have to be kept fixed and cannot be changed without reparameterizing the entire model. They 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.

5.22.3 Description

The *oxdna/fene*, *oxdna2/fene*, and *oxrna2/fene* bond styles use the potential

$$E = -\frac{\epsilon}{2} \ln \left[1 - \left(\frac{r - r_0}{\Delta} \right)^2 \right]$$

to define a modified finite extensible nonlinear elastic (FENE) potential *(Ouldridge)* to model the connectivity of the phosphate backbone in the oxDNA/oxRNA force field for coarse-grained modelling of DNA/RNA.

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

- ϵ (energy)
- Δ (distance)
- r_0 (distance)

Note

The oxDNA bond style has to be used together with the corresponding oxDNA pair styles for excluded volume interaction *oxdna/excv* , stacking *oxdna/stk* , cross-stacking *oxdna/xstk* and coaxial stacking interaction *oxdna/coaxstk* as well as hydrogen-bonding interaction *oxdna/hbond* (see also documentation of *pair_style oxdna/excv*). For the oxDNA2 *(Snodin)* bond style the analogous pair styles *oxdna2/excv* , *oxdna2/stk* , *oxdna2/xstk* , *oxdna2/coaxstk* , *oxdna2/hbond* and an additional Debye-Hueckel pair style *oxdna2/dh* have to be defined. The same applies to the oxRNA2 *(Sulc1)* styles.

Note

This bond style has 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.

Warning

If data files are produced with *write_data*, then the *newton* command should be set to *newton on* or *newton off on*. Otherwise the data files will not have the same 3'-to-5' polarity as the initial data file. This limitation does not apply to binary restart files produced with *write_restart*.

Example input and data files for DNA and RNA duplexes can be found in examples/PACKAGES/cgdna/examples/oxDNA/`, `.../oxDNA2/ and .../oxRNA2/. A simple python setup tool which creates single straight or helical DNA strands, DNA/RNA duplexes or arrays of DNA/RNA 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/oxRNA publications. These are *(Ouldridge)* and *(Ouldridge-DPhil)* for oxDNA, *(Snodin)* for oxDNA2, *(Sulc1)* for oxRNA2 and for sequence-specific hydrogen-bonding and stacking interactions *(Sulc2)*.

5.22.4 Potential file reading

For each style oxdna, oxdna2 and oxrna2, the first parameter argument can be a filename, and if it is, no further arguments should be supplied. Therefore the following command:

```
bond_style oxdna/fene
bond_coeff * oxdna_lj.cgdna
```

will be interpreted as a request to read the (FENE) potential *(Ouldridge)* parameters from the file with the given name. The file can define multiple potential parameters for both bonded and pair interactions, but for the above bonded interactions there must exist in the file a line of the form:

```
*   fene    epsilon delta r0
```

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/.

5.22.5 Restrictions

This bond style 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.

5.22.6 Related commands

pair_style oxdna/excv, *pair_style oxdna2/excv*, *pair_style oxrna2/excv*, *bond_coeff*, *atom_style oxdna*, *fix nve/dotc/langevin*

5.22.7 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).

(Snodin) B.E. Snodin, F. Randisi, M. Mosayebi, et al., J. Chem. Phys. 142, 234901 (2015).

(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).

5.23 bond_style quartic command

Accelerator Variants: *quartic/omp*

5.23.1 Syntax

```
bond_style quartic
```

5.23.2 Examples

```
bond_style quartic
bond_coeff 2 1200 -0.55 0.25 1.3 34.6878
```

5.23.3 Description

The *quartic* bond style uses the potential

$$\begin{aligned}
E &= E_q + E_{LJ} \\
E_q &= K(r - R_c)^2(r - R_c - B_1)(r - R_c - B_2) + U_0 \\
E_{LJ} &= \begin{cases} 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] + \epsilon & : \quad r < 2^{\frac{1}{6}}, \epsilon = 1, \sigma = 1 \\ 0 & : \quad r >= 2^{\frac{1}{6}} \end{cases}
\end{aligned}$$

to define a bond that can be broken as the simulation proceeds (e.g. due to a polymer being stretched). The σ and ϵ used in the LJ portion of the formula are both set equal to 1.0 by LAMMPS and the LJ portion is cut off at its minimum, i.e. at $r_c = 2^{\frac{1}{6}}$.

The following coefficients must be defined for each bond type via the *bond_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/distance^4)
- B_1 (distance)
- B_2 (distance)
- R_c (distance)
- U_0 (energy)

This potential was constructed to mimic the FENE bond potential for coarse-grained polymer chains. When monomers with $\sigma = \epsilon = 1.0$ are used, the following choice of parameters gives a quartic potential that looks nearly like the FENE potential:

$$\begin{aligned}
K &= 1200 \\
B_1 &= -0.55 \\
B_2 &= 0.25 \\
R_c &= 1.3 \\
U_0 &= 34.6878
\end{aligned}$$

Different parameters can be specified using the *bond_coeff* command, but you will need to choose them carefully so they form a suitable bond potential.

R_c is the cutoff length at which the bond potential goes smoothly to a local maximum. If a bond length ever becomes $> R_c$, LAMMPS "breaks" the bond, which means two things. First, the bond potential is turned off by setting its type to 0, and is no longer computed. Second, a pairwise interaction between the two atoms is turned on, since they are no longer bonded. See the *Howto* page on broken bonds for more information.

LAMMPS does the second task via a computational sleight-of-hand. It subtracts the pairwise interaction as part of the bond computation. When the bond breaks, the subtraction stops. For this to work, the pairwise interaction must always be computed by the *pair_style* command, whether the bond is broken or not. This means that *special_bonds* must be set to 1,1,1, as indicated as a restriction below.

Note that when bonds are dumped to a file via the *dump local* command, bonds with type 0 are not included. The *delete_bonds* command can also be used to query the status of broken bonds or permanently delete them, e.g.:

```
delete_bonds all stats
delete_bonds all bond 0 remove
```

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.

5.23.4 Restrictions

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

The *quartic* style requires that *special_bonds* parameters be set to 1,1,1. Three- and four-body interactions (angle, dihedral, etc) cannot be used with *quartic* bonds.

5.23.5 Related commands

bond_coeff, *delete_bonds*

5.23.6 Default

none

5.24 bond_style rheo/shell command

5.24.1 Syntax

```
bond_style rheo/shell keyword value attribute1 attribute2 ...
```

- required keyword = *t/form*
- optional keyword = *store/local*

```
t/form value = formation time for a bond (time units)

store/local values = fix_ID N attributes ...
  * fix_ID = ID of associated internal fix to store data
  * N = prepare data for output every this many timesteps
  * attributes = zero or more of the below attributes may be appended

    id1, id2 = IDs of two atoms in the bond
    time = the timestep the bond broke
    x, y, z = the center of mass position of the two atoms when the bond broke (distance units)
    x/ref, y/ref, z/ref = the initial center of mass position of the two atoms (distance units)
```

5.24.2 Examples

```
bond_style rheo/shell t/form 10.0
bond_coeff 1 1.0 0.05 0.1
```

5.24.3 Description

Added in version 29Aug2024.

The *rheo/shell* bond style is designed to work with *fix rheo/oxidation* which creates candidate bonds between eligible surface or near-surface particles. When a bond is first created, it computes no forces and starts a timer. Forces are not computed until the timer reaches the specified bond formation time, *t/form*, and the bond is enabled and applies forces. If the two particles move outside of the maximum bond distance or move into the bulk before the timer reaches *t/form*, the bond automatically deletes itself. This deletion is not recorded as a broken bond in the optional *store/local* fix.

Before bonds are enabled, they are still treated as regular bonds by all other parts of LAMMPS. This means they are written to data files and counted in computes such as *nbond/atom*. To only count enabled bonds, use the *nbond/shell* attribute in *compute rheo/property/atom*.

When enabled, the bond then computes forces based on deviations from the initial reference state of the two atoms much like a BPM style bond (as further discussed in the *BPM howto page*). The reference state is stored by each bond when it is first enabled. Data is then preserved across run commands and is written to *binary restart files* such that restarting the system will not reset the reference state of a bond or the timer.

This bond style is based on a model described in *(Clemmer)*. The force has a magnitude of

$$F = 2k(r - r_0) + \frac{2k}{r_0^2 \epsilon_c^2}(r - r_0)^3$$

where k is a stiffness, r is the current distance and r_0 is the initial distance between the two particles, and ϵ_c is maximum strain beyond which a bond breaks. This is done by setting the bond type to 0 such that forces are no longer computed.

A damping force proportional to the difference in the normal velocity of particles is also applied to bonded particles:

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

where γ is the damping strength, $\hat{r}$ is the displacement normal vector, and $\vec{v}$ is the velocity difference between the two particles.

The following coefficients must be defined for each bond type via the *bond_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 (force/distance units)
- ϵ_c (unit less)
- γ (force/velocity units)

Unlike other BPM-style bonds, this bond style does not update special bond settings when bonds are created or deleted. This bond style also does not enforce specific *special_bonds* settings. This behavior is purposeful such *RHEO pair* forces and heat flows are still calculated.

If the *store/local* keyword is used, an internal fix will track bonds that break during the simulation. Whenever a bond breaks, data is processed and transferred to an internal fix labeled *fix_ID*. This allows the local data to be accessed by other LAMMPS commands. Following this optional keyword, a list of one or more attributes is specified. These include the IDs of the two atoms in the bond. The other attributes for the two atoms include the timestep during which the bond broke and the current/initial center of mass position of the two atoms.

Data is continuously accumulated 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.

Note that when unbroken bonds are dumped to a file via the *dump local* command, bonds with type 0 (broken bonds) are not included. The *delete_bonds* command can also be used to query the status of broken bonds or permanently delete them, e.g.:

```
delete_bonds all stats
delete_bonds all bond 0 remove
```

5.24.4 Restart and other info

This bond style writes the reference state of each bond to *binary restart files*. Loading a restart file will properly restore bonds. However, the reference state is NOT written to data files. Therefore reading a data file will not restore bonds and will cause their reference states to be redefined.

If the *store/local* option is used, an internal fix will calculate 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, broken bonds. 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.

The single() function of this bond style returns 0.0 for the energy of a bonded interaction, since energy is not conserved in these dissipative potentials. The single() function also calculates two extra bond quantities, the initial distance r_0 and a time. These extra quantities can be accessed by the *compute bond/local* command as *b1* and *b2*.

5.24.5 Restrictions

This bond 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.

5.24.6 Related commands

bond_coeff, *fix rheo/oxidation*

5.24.7 Default

NA

(Clemmer) Clemmer, Pierce, O'Connor, Nevins, Jones, Lechman, Tencer, Appl. Math. Model., 130, 310-326 (2024).

5.25 bond_style special command

5.25.1 Syntax

```
bond_style special
```

5.25.2 Examples

```
bond_style special
bond_coeff 0.5 0.5
```

5.25.3 Description

The *special* bond style can be used to create conceptual bonds which effectively impose weightings on the pairwise Lennard Jones and/or Coulombic interactions between selected pairs of particles in the system. The form of the pairwise interaction will be whatever is computed by the *pair_style* command defined for the system; this command defines the weightings for its two terms.

This command can thus be useful to apply weightings that cannot be handled by the *special_bonds* command, such as on 1-5 or 1-6 interactions. Or it can be used to add pairwise forces between one or more pairs of atoms that otherwise would not be include in the *pair_style* computation.

The potential for this bond style has the form

$$E = w_{LJ}E_{LJ} + w_{Coul}E_{Coul}$$

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

- w_{LJ} weight (0.0 to 1.0) on pairwise Lennard-Jones interactions
- w_{Coul} weight (0.0 to 1.0) on pairwise Coulombic interactions

---

Normally this bond style should be used in conjunction with one (or more) other bond styles which compute forces between atoms directly bonded to each other in a molecule. This means the *bond_style hybrid* command should be used with bond_style special as one of its sub-styles.

Note that the same as for any other bond style, pairs of bonded atoms must be enumerated in the data file read by the *read_data* command. Thus if this command is used to weight all 1-5 interactions in the system, all the 1-5 pairs of atoms must be listed in the "Bonds" section of the data file.

This bond style imposes strict requirements on settings made with the *special_bonds* command. These requirements ensure that the new bonds created by this style do not create spurious 1-2, 1-3, or 1-4 interactions within the molecular topology.

Specifically 1-2 interactions must have weights of zero, 1-3 interactions must either have weights of unity or *special_bonds angle yes* must be used, and 1-4 interactions must have weights of unity or *special_bonds dihedral yes* must be used.

If this command is used to create bonded interactions between particles that are further apart than usual (e.g. 1-5 or 1-6 interactions), this style may require an increase in the communication cutoff via the *comm_modify cutoff* command. If LAMMPS cannot find a partner atom in a bond, an error will be issued.

5.25.4 Restrictions

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

This bond style requires the use of a *pair_style* which computes a pairwise additive interaction and provides the ability to compute interactions for individual pairs of atoms. Manybody potentials are not compatible in general, but also some other pair styles are missing the required functionality and thus will cause an error.

This command is not compatible with long-range Coulombic interactions. If a *kspace_style <kspace_style>* is declared, an error will be issued.

5.25.5 Related commands

bond_coeff, *special_bonds*

5.25.6 Default

none

5.26 bond_style table command

Accelerator Variants: *table/omp*

5.26.1 Syntax

```
bond_style table style N
```

- style = *linear* or *spline* = method of interpolation
- N = use N values in table

5.26.2 Examples

```
bond_style table linear 1000
bond_coeff 1 file.table ENTRY1
```

5.26.3 Description

Style *table* creates interpolation tables of length *N* from bond potential and force values listed in a file(s) as a function of bond length. The files are read by the *bond_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 2 styles: *linear* or *spline*.

For the *linear* style, the bond length 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 bond length is used to find the appropriate set of coefficients which are used to evaluate a cubic polynomial which computes the energy or force.

The following coefficients must be defined for each bond type via the *bond_coeff* command as in the example above.

- filename
- keyword

The filename specifies a file containing tabulated energy and force values. The keyword specifies a section of the file. The format of this file is described below.

Suitable tables for use with this bond style can be created by LAMMPS itself from existing bond styles using the *bond_write* command. This can be useful to have a template file for testing the bond style settings and to build a compatible custom file. 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 is as follows (without the parenthesized comments):

```
# Bond potential for harmonic (one or more comment or blank lines)

HAM                          (keyword is the first text on line)
N 101 FP 0 0 EQ 0.5          (N, FP, EQ  parameters)
                             (blank line)
1 0.00 338.0000 1352.0000    (index, bond-length, energy, force)
2 0.01 324.6152 1324.9600
...
101 1.00 338.0000 -1352.0000
```

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 *bond_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 *bond_style table* command. Let Ntable = *N* in the bond_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 bond lengths. 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.

The "FP" parameter is optional. If used, it is followed by two values fplo and fphi, which are the derivatives of the force at the innermost and outermost bond lengths. 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 force values in the table.

The "EQ" parameter is also optional. If used, it is followed by a the equilibrium bond length, which is used, for example, by the *fix shake* command. If not used, the equilibrium bond length is to the distance in the table with the lowest potential energy.

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 bond length r (in distance units), the third value is the energy (in energy units), and the fourth is the force (in force units). The bond lengths must range from a LO value to a HI value, and increase from one line to the next. If the actual bond length is ever smaller than the LO value or larger than the HI value, then the calculation is aborted with an error, so it is advisable to cover the whole range of possible bond lengths.

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.

5.26.4 Restart info

This bond style writes the settings for the "bond_style table" command to *binary restart files*, so a bond_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, bond_coeff commands do need to be specified in the restart input script.

5.26.5 Restrictions

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

5.26.6 Related commands

bond_coeff, *delete_bonds*, *bond_write*

5.26.7 Default

none

5.27 bond_style zero command

5.27.1 Syntax

```
bond_style zero keyword
```

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

5.27.2 Examples

```
bond_style zero
bond_style zero nocoeff
bond_coeff *
bond_coeff * 2.14
```

5.27.3 Description

Using an bond style of zero means bond forces and energies are not computed, but the geometry of bond pairs is still accessible to other commands.

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

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

Note that the *bond_coeff* command must be used for all bond types. If specified, there can be only one value, which is going to be used to assign an equilibrium distance, e.g. for use with *fix shake*.

5.27.4 Restrictions

none

5.27.5 Related commands

bond_style none

5.27.6 Default

none