CHAPTER

FIVE

COMMANDS

These pages describe how a LAMMPS input script is formatted and the commands in it are used to define a LAMMPS simulation.

5.1 LAMMPS input scripts

LAMMPS executes calculations by reading commands from a input script (text file), one line at a time. When the input script ends, LAMMPS exits. This is different from programs that read and process the entire input before starting a calculation.

Each command causes LAMMPS to take some immediate action without regard for any commands that may be processed later. Commands may set an internal variable, read in a file, or run a simulation. These actions can be grouped into three categories:

a) commands that change a global setting (examples: *timestep*, *newton*, *echo*, *log*, *thermo*, *restart*),

b) commands that add, modify, remove, or replace "styles" that are executed during a "run" (examples: *pair_style*, *fix*, *compute*, *dump*, *thermo_style*, *pair_modify*), and

c) commands that execute a "run" or perform some other computation or operation (examples: *print*, *run*, *minimize*, *temper*, *write_dump*, *rerun*, *read_data*, *read_restart*)

Commands in category a) have default settings, which means you only need to use the command if you wish to change the defaults.

In many cases, the ordering of commands in an input script is not important, but can have consequences when the global state is changed between commands in the c) category. The following rules apply:

(1) LAMMPS does not read your entire input script and then perform a simulation with all the settings. Rather, the input script is read one line at a time and each command takes effect when it is read. Thus this sequence of commands:

```
timestep 0.5
run      100
run      100
```

does something different than this sequence:

```
run      100
timestep 0.5
run      100
```

In the first case, the specified timestep (0.5 fs) is used for two simulations of 100 timesteps each. In the second case, the default timestep (1.0 fs) is used for the first 100 step simulation and a 0.5 fs timestep is used for the second one.

(2) Some commands are only valid when they follow other commands. For example you cannot set the temperature of a group of atoms until atoms have been defined and a group command is used to define which atoms belong to the group.

(3) Sometimes command B will use values that can be set by command A. This means command A must precede command B in the input script if it is to have the desired effect. For example, the *read_data* command initializes the system by setting up the simulation box and assigning atoms to processors. If default values are not desired, the *processors* and *boundary* commands need to be used before read_data to tell LAMMPS how to map processors to the simulation box.

Many input script errors are detected by LAMMPS and an ERROR or WARNING message is printed. The *Errors* page gives more information on what errors mean. The documentation for each command lists restrictions on how the command can be used.

You can use the *-skiprun* command line flag to have LAMMPS skip the execution of any run, minimize, or similar commands to check the entire input for correct syntax to avoid crashes on typos or syntax errors in long runs.

5.2 Parsing rules for input scripts

Each non-blank line in the input script is treated as a command. LAMMPS commands are case sensitive. Command names are lower-case, as are specified command arguments. Upper case letters may be used in file names or user-chosen ID strings.

Here are 6 rules for how each line in the input script is parsed by LAMMPS:

1. If the last printable character on the line is a "&" character, the command is assumed to continue on the next line. The next line is concatenated to the previous line by removing the "&" character and line break. This allows long commands to be continued across two or more lines. See the discussion of triple quotes in *6* for how to continue a command across multiple line without using "&" characters.

2. All characters from the first "#" character onward are treated as comment and discarded. The exception to this rule is described in *6*. Note that a comment after a trailing "&" character will prevent the command from continuing on the next line. Also note that for multi-line commands a single leading "#" will comment out the entire command.

```
# this is a comment
timestep  1.0   # this is also a comment
```

3. The line is searched repeatedly for $ characters, which indicate variables that are replaced with a text string. The exception to this rule is described in *6*.

   If the $ is followed by text in curly brackets '{}', then the variable name is the text inside the curly brackets. If no curly brackets follow the $, then the variable name is the single character immediately following the $. Thus ${myTemp} and $x refer to variables named "myTemp" and "x", while $xx will be interpreted as a variable named "x" followed by an "x" character.

   How the variable is converted to a text string depends on what style of variable it is; see the *variable* page for details. It can be a variable that stores multiple text strings, and return one of them. The returned text string can be multiple "words" (space separated) which will then be interpreted as multiple arguments in the input command. The variable can also store a numeric formula which will be evaluated and its numeric result returned as a string.

As a special case, if the $ is followed by parenthesis "()", then the text inside the parenthesis is treated as an "immediate" variable and evaluated as an *equal-style variable*. This is a way to use numeric formulas in an input script without having to assign them to variable names. For example, these 3 input script lines:

```
variable X equal (xlo+xhi)/2+sqrt(v_area)
region 1 block $X 2 INF INF EDGE EDGE
variable X delete
```

can be replaced by:

```
region 1 block $((xlo+xhi)/2+sqrt(v_area)) 2 INF INF EDGE EDGE
```

so that you do not have to define (or discard) a temporary variable, "X" in this case.

Additionally, the entire "immediate" variable expression may be followed by a colon, followed by a C-style format string, e.g. :%f or :%.10g. The format string must be appropriate for a double-precision floating-point value. The format string is used to output the result of the variable expression evaluation. If a format string is not specified, a high-precision %.20g is used as the default format.

This can be useful for formatting print output to a desired precision:

```
print "Final energy per atom: $(v_ke_per_atom+v_pe_per_atom:%10.3f) eV/atom"
```

Note that neither the curly-bracket or immediate form of variables can contain nested $ characters for other variables to substitute for. Thus you may **NOT** do this:

```
variable      a equal 2
variable      b2 equal 4
print         "B2 = ${b$a}"
```

Nor can you specify an expression like $($x-1.0) for an immediate variable, but you could use $(v_x-1.0), since the latter is valid syntax for an *equal-style variable*.

See the *variable* command for more details of how strings are assigned to variables and evaluated, and how they can be used in input script commands.

4. The line is broken into "words" separated by white-space (tabs, spaces). Note that words can thus contain letters, digits, underscores, or punctuation characters.

5. The first word is the command name. All successive words in the line are arguments.

6. If you want text with spaces to be treated as a single argument, it can be enclosed in either single (') or double (") or triple ("""") quotes. A long single argument enclosed in single or double quotes can span multiple lines if the "&" character is used, as described in *1* above. When the lines are concatenated together by LAMMPS (and the "&" characters and line breaks removed), the combined text will become a single line. If you want multiple lines of an argument to retain their line breaks, the text can be enclosed in triple quotes, in which case "&" characters are not needed and do not function as line continuation character. For example:

```
print "Volume = $v"
print 'Volume = $v'
if "${steps} > 1000" then quit
variable a string "red green blue &
                   purple orange cyan"
print """
System volume = $v
System temperature = $t
"""
```

In each of these cases, the single, double, or triple quotes are removed and the enclosed text stored internally as a single argument.

See the *dump modify format*, *print*, *if*, and *python* commands for examples.

A "#" or "$" character that is between quotes will not be treated as a comment indicator in *2* or substituted for as a variable in *3*.

Note

If the argument is itself a command that requires a quoted argument (e.g. using a *print* command as part of an *if* or *run every* command), then single, double, or triple quotes can be nested in the usual manner. See the doc pages for those commands for examples. Only one of level of nesting is allowed, but that should be sufficient for most use cases.

ASCII versus UTF-8

LAMMPS expects and processes 7-bit ASCII format text internally. Many modern environments use UTF-8 encoding, which is a superset of the 7-bit ASCII character table and thus mostly compatible. However, there are several non-ASCII characters that can look very similar to their ASCII equivalents or are invisible (so they look like a blank), but are encoded differently. Web browsers, PDF viewers, document editors are known to sometimes replace one with the other for a better looking output. However, that can lead to problems, for instance, when using cut-n-paste of input file examples from web pages, or when using a document editor (not a dedicated plain text editor) for writing LAMMPS inputs. LAMMPS will try to detect this and substitute the non-ASCII characters with their ASCII equivalents where known. There also is going to be a warning printed, if this occurs. It is recommended to avoid such characters altogether in LAMMPS input, data and potential files. The replacement tables are likely incomplete and dependent on users reporting problems processing correctly looking input containing UTF-8 encoded non-ASCII characters.

5.3 Input script structure

This page describes the structure of a typical LAMMPS input script. The examples directory in the LAMMPS distribution contains many sample input scripts; it is discussed on the *Examples* doc page.

A LAMMPS input script typically has 4 parts:

1. *Initialization*
2. *System definition*
3. *Simulation settings*
4. *Run a simulation*

The last 2 parts can be repeated as many times as desired. I.e. run a simulation, change some settings, run some more, etc. Each of the 4 parts is now described in more detail. Remember that almost all commands need only be used if a non-default value is desired.

5.3.1 Initialization

Set parameters that need to be defined before atoms are created or read-in from a file.

The relevant commands are *units*, *dimension*, *newton*, *processors*, *boundary*, *atom_style*, *atom_modify*.

If force-field parameters appear in the files that will be read, these commands tell LAMMPS what kinds of force fields are being used: *pair_style*, *bond_style*, *angle_style*, *dihedral_style*, *improper_style*.

5.3.2 System definition

There are 3 ways to define the simulation cell and reserve space for force field info and fill it with atoms in LAMMPS. Read them in from (1) a data file or (2) a restart file via the *read_data* or *read_restart* commands, respectively. These files can also contain molecular topology information. Or (3) create a simulation cell and fill it with atoms on a lattice (with no molecular topology), using these commands: *lattice*, *region*, *create_box*, *create_atoms* or *read_dump*.

The entire set of atoms can be duplicated to make a larger simulation using the *replicate* command.

5.3.3 Simulation settings

Once atoms and molecular topology are defined, a variety of settings can be specified: force field coefficients, simulation parameters, output options, and more.

Force field coefficients are set by these commands (they can also be set in the read-in files): *pair_coeff*, *bond_coeff*, *angle_coeff*, *dihedral_coeff*, *improper_coeff*, *kspace_style*, *dielectric*, *special_bonds*.

Various simulation parameters are set by these commands: *neighbor*, *neigh_modify*, *group*, *timestep*, *reset_timestep*, *run_style*, *min_style*, *min_modify*.

Fixes impose a variety of boundary conditions, time integration, and diagnostic options. The *fix* command comes in many flavors.

Various computations can be specified for execution during a simulation using the *compute*, *compute_modify*, and *variable* commands.

Output options are set by the *thermo*, *dump*, and *restart* commands.

5.3.4 Run a simulation

A molecular dynamics simulation is run using the *run* command. Energy minimization (molecular statics) is performed using the *minimize* command. A parallel tempering (replica-exchange) simulation can be run using the *temper* command.

5.4 Commands by category

This page lists most of the LAMMPS commands, grouped by category. The *General commands* page lists all general commands alphabetically. Style options for entries like fix, compute, pair etc. have their own pages where they are listed alphabetically.

5.4.1 Initialization

newton	*package*	*processors*	*suffix*	*units*

5.4.2 Setup simulation box

boundary	*change_box*	*create_box*	*dimension*
lattice	*region*		

5.4.3 Setup atoms

atom_modify	*atom_style*	*balance*	*create_atoms*
create_bonds	*delete_atoms*	*delete_bonds*	*displace_atoms*
group	*mass*	*molecule*	*read_data*
read_dump	*read_restart*	*replicate*	*set*
velocity			

5.4.4 Force fields

angle_coeff	*angle_style*	*bond_coeff*	*bond_style*
bond_write	*dielectric*	*dihedral_coeff*	*dihedral_style*
improper_coeff	*improper_style*	*kspace_modify*	*kspace_style*
pair_coeff	*pair_modify*	*pair_style*	*pair_write*
special_bonds			

5.4.5 Settings

comm_modify	*comm_style*	*info*	*min_modify*
min_style	*neigh_modify*	*neighbor*	*partition*
reset_timestep	*run_style*	*timer*	*timestep*

5.4.6 Operations within timestepping (fixes) and diagnostics (computes)

compute	*compute_modify*	*fix*	*fix_modify*
uncompute	*unfix*		

5.4.7 Output

dump image	*dump movie*	*dump*	*dump_modify*
restart	*thermo*	*thermo_modify*	*thermo_style*
undump	*write_coeff*	*write_data*	*write_dump*
write_restart			

5.4.8 Actions

minimize	*neb*	*neb_spin*	*prd*	*rerun*	*run*
tad	*temper*				

5.4.9 Input script control

clear	*echo*	*if*	*include*	*info*	*jump*	*label*
log	*next*	*print*	*python*	*quit*	*shell*	*variable*

5.5 General commands

An alphabetic list of general LAMMPS commands.

angle_coeff	*angle_style*	*angle_write*	*atom_modify*	*atom_style*	*balance*
bond_coeff	*bond_style*	*bond_write*	*boundary*	*change_box*	*clear*
comm_modify	*comm_style*	*compute*	*compute_modify*	*create_atoms*	*create_bonds*
create_box	*delete_atoms*	*delete_bonds*	*dielectric*	*dihedral_coeff*	*dihedral_style*
dihedral_write	*dimension*	*displace_atoms*	*dump*	*dump_modify*	*echo*
fix	*fix_modify*	*geturl*	*group*	*if*	*improper_coeff*
improper_style	*include*	*info*	*jump*	*kspace_modify*	*kspace_style*
label	*labelmap*	*lattice*	*log*	*mass*	*minimize*
min_modify	*min_style*	*molecule*	*neigh_modify*	*neighbor*	*newton*
next	*package*	*pair_coeff*	*pair_modify*	*pair_style*	*pair_write*
partition	*print*	*processors*	*quit*	*read_data*	*read_dump*
read_restart	*region*	*replicate*	*rerun*	*reset_atoms*	*reset_timestep*
restart	*run*	*run_style*	*set*	*shell*	*special_bonds*
suffix	*thermo*	*thermo_modify*	*thermo_style*	*timer*	*timestep*
uncompute	*undump*	*unfix*	*units*	*variable*	*velocity*
write_coeff	*write_data*	*write_dump*	*write_restart*		

Additional general LAMMPS commands provided by packages. A few commands have accelerated versions. This is indicated by an additional letter in parenthesis: k = KOKKOS.

dynamical_matrix (k)	*group2ndx*	*hyper*	*kim*	*fitpod*	*mdi*
ndx2group	*neb*	*neb/spin*	*plugin*	*prd*	*python*
tad	*temper*	*temper/grem*	*temper/npt*	*third_order (k)*	

5.6 Fix styles

An alphabetic list of all LAMMPS *fix* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

accelerate/cos	*acks2/reaxff (k)*	*adapt*	*adapt/fep*
addforce	*add/heat*	*addtorque*	*alchemy*
amoeba/bitorsion	*amoeba/pitorsion*	*append/atoms*	*atc*
atom/swap	*ave/atom*	*ave/chunk*	*ave/correlate*
ave/correlate/long	*ave/grid*	*ave/histo*	*ave/histo/weight*
ave/time	*aveforce*	*balance*	*bocs*
bond/break	*bond/create*	*bond/create/angle*	*bond/react*
bond/swap	*box/relax*	*brownian*	*brownian/asphere*
brownian/sphere	*charge/regulation*	*cmap*	*colvars*
controller	*damping/cundall*	*deform (k)*	*deform/pressure*
deposit	*dpd/energy (k)*	*drag*	*drude*
drude/transform/direct	*drude/transform/inverse*	*dt/reset (k)*	*edpd/source*
efield (k)	*efield/tip4p*	*ehex*	*electrode/conp (i)*
electrode/conq (i)	*electrode/thermo (i)*	*electron/stopping*	*electron/stopping/fit*
enforce2d (k)	*eos/cv*	*eos/table*	*eos/table/rx (k)*

continues on next page

Table 1 – continued from previous page

evaporate	*external*	*ffl*	*filter/corotate*
flow/gauss	*freeze (k)*	*gcmc*	*gld*
gle	*gravity (ko)*	*grem*	*halt*
heat	*heat/flow*	*hyper/global*	*hyper/local*
imd	*indent*	*ipi*	*langevin (k)*
langevin/drude	*langevin/eff*	*langevin/spin*	*lb/fluid*
lb/momentum	*lb/viscous*	*lineforce*	*manifoldforce*
mdi/qm	*mdi/qmmm*	*meso/move*	*mol/swap*
momentum (k)	*momentum/chunk*	*move*	*msst*
mvv/dpd	*mvv/edpd*	*mvv/tdpd*	*neb*
neb/spin	*nonaffine/displacement*	*nph (ko)*	*nph/asphere (o)*
nph/body	*nph/eff*	*nph/sphere (o)*	*nphug*
npt (giko)	*npt/asphere (o)*	*npt/body*	*npt/cauchy*
npt/eff	*npt/sphere (o)*	*npt/uef*	*numdiff*
numdiff/virial	*nve (giko)*	*nve/asphere (gi)*	*nve/asphere/noforce*
nve/awpmd	*nve/body*	*nve/dot*	*nve/dotc/langevin*
nve/eff	*nve/limit*	*nve/line*	*nve/manifold/rattle*
nve/noforce	*nve/sphere (ko)*	*nve/bpm/sphere*	*nve/spin*
nve/tri	*nvk*	*nvt (giko)*	*nvt/asphere (o)*
nvt/body	*nvt/eff*	*nvt/manifold/rattle*	*nvt/sllod (iko)*
nvt/sllod/eff	*nvt/sphere (o)*	*nvt/uef*	*oneway*
orient/bcc	*orient/fcc*	*orient/eco*	*pafi*
pair	*phonon*	*pimd/langevin*	*pimd/nvt*
planeforce	*plumed*	*poems*	*polarize/bem/gmres*
polarize/bem/icc	*polarize/functional*	*pour*	*precession/spin*
press/berendsen	*press/langevin*	*print*	*propel/self*
property/atom (k)	*python/invoke*	*python/move*	*qbmsst*
qeq/comb (o)	*qeq/dynamic*	*qeq/fire*	*qeq/point*
qeq/reaxff (ko)	*qeq/shielded*	*qeq/slater*	*qmmm*
qtb	*rattle*	*reaxff/bonds (k)*	*reaxff/species (k)*
recenter	*restrain*	*rheo*	*rheo/oxidation*
rheo/pressure	*rheo/thermal*	*rheo/viscosity*	*rhok*
rigid (o)	*rigid/meso*	*rigid/nph (o)*	*rigid/nph/small*
rigid/npt (o)	*rigid/npt/small*	*rigid/nve (o)*	*rigid/nve/small*
rigid/nvt (o)	*rigid/nvt/small*	*rigid/small (o)*	*rx (k)*
saed/vtk	*setforce (k)*	*setforce/spin*	*sgcmc*
shake (k)	*shardlow (k)*	*smd*	*smd/adjust_dt*
smd/integrate_tlsph	*smd/integrate_ulsph*	*smd/move_tri_surf*	*smd/setvel*
smd/wall_surface	*sph*	*sph/stationary*	*spring*
spring/chunk	*spring/rg*	*spring/self (k)*	*srd*
store/force	*store/state*	*tdpd/source*	*temp/berendsen (k)*
temp/csld	*temp/csvr*	*temp/rescale (k)*	*temp/rescale/eff*
tfmc	*tgnpt/drude*	*tgnvt/drude*	*thermal/conductivity*
ti/spring	*tmd*	*ttm*	*ttm/grid*
ttm/mod	*tune/kspace*	*vector*	*viscosity*
viscous (k)	*viscous/sphere*	*wall/body/polygon*	*wall/body/polyhedron*
wall/colloid	*wall/ees*	*wall/flow (k)*	*wall/gran (k)*
wall/gran/region	*wall/harmonic*	*wall/lj1043*	*wall/lj126*
wall/lj93 (k)	*wall/lepton*	*wall/morse*	*wall/piston*
wall/reflect (k)	*wall/reflect/stochastic*	*wall/region*	*wall/region/ees*
wall/srd	*wall/table*	*widom*	

5.7 Compute styles

An alphabetic list of all LAMMPS *compute* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

ackland/atom	*adf*	*aggregate/atom*	*angle*
angle/local	*angmom/chunk*	*ave/sphere/atom (k)*	*basal/atom*
body/local	*bond*	*bond/local*	*born/matrix*
centro/atom	*centroid/stress/atom*	*chunk/atom*	*chunk/spread/atom*
cluster/atom	*cna/atom*	*cnp/atom*	*com*
com/chunk	*contact/atom*	*coord/atom (k)*	*count/type*
damage/atom	*dihedral*	*dihedral/local*	*dilatation/atom*
dipole	*dipole/chunk*	*dipole/tip4p*	*dipole/tip4p/chunk*
displace/atom	*dpd*	*dpd/atom*	*edpd/temp/atom*
efield/atom	*efield/wolf/atom*	*entropy/atom*	*erotate/asphere*
erotate/rigid	*erotate/sphere (k)*	*erotate/sphere/atom*	*event/displace*
fabric	*fep*	*fep/ta*	*force/tally*
fragment/atom	*global/atom*	*group/group*	*gyration*
gyration/chunk	*gyration/shape*	*gyration/shape/chunk*	*heat/flux*
heat/flux/tally	*heat/flux/virial/tally*	*hexorder/atom*	*hma*
improper	*improper/local*	*inertia/chunk*	*ke*
ke/atom	*ke/atom/eff*	*ke/eff*	*ke/rigid*
composition/atom (k)	*mliap*	*momentum*	*msd*
msd/chunk	*msd/nongauss*	*nbond/atom*	*omega/chunk*
orientorder/atom (k)	*pace*	*pair*	*pair/local*
pe	*pe/atom*	*pe/mol/tally*	*pe/tally*
plasticity/atom	*pod/atom*	*podd/atom*	*pod/local*
pod/global	*pressure*	*pressure/alchemy*	*pressure/uef*
property/atom	*property/chunk*	*property/grid*	*property/local*
ptm/atom	*rattlers/atom*	*rdf*	*reaxff/atom (k)*
reduce	*reduce/chunk*	*reduce/region*	*rheo/property/atom*
rigid/local	*saed*	*slcsa/atom*	*slice*
smd/contact/radius	*smd/damage*	*smd/hourglass/error*	*smd/internal/energy*
smd/plastic/strain	*smd/plastic/strain/rate*	*smd/rho*	*smd/tlsph/defgrad*
smd/tlsph/dt	*smd/tlsph/num/neighs*	*smd/tlsph/shape*	*smd/tlsph/strain*
smd/tlsph/strain/rate	*smd/tlsph/stress*	*smd/triangle/vertices*	*smd/ulsph/effm*
smd/ulsph/num/neighs	*smd/ulsph/strain*	*smd/ulsph/strain/rate*	*smd/ulsph/stress*
smd/vol	*snap*	*sna/atom*	*sna/grid*
sna/grid/local	*snad/atom*	*snav/atom*	*sph/e/atom*
sph/rho/atom	*sph/t/atom*	*spin*	*stress/atom*
stress/cartesian	*stress/cylinder*	*stress/mop*	*stress/mop/profile*
stress/spherical	*stress/tally*	*tdpd/cc/atom*	*temp (k)*
temp/asphere	*temp/body*	*temp/chunk*	*temp/com*
temp/cs	*temp/deform (k)*	*temp/deform/eff*	*temp/drude*
temp/eff	*temp/partial*	*temp/profile*	*temp/ramp*
temp/region	*temp/region/eff*	*temp/rotate*	*temp/sphere*
temp/uef	*ti*	*torque/chunk*	*vacf*
vcm/chunk	*viscosity/cos*	*voronoi/atom*	*xrd*

5.8 Pair styles

All LAMMPS *pair_style* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

none	*zero*	*hybrid (ko)*
hybrid/molecular (o)	*hybrid/overlay (ko)*	*hybrid/scaled (o)*
kim	*list*	*tracker*
adp (ko)	*agni (o)*	*aip/water/2dm (t)*
airebo (io)	*airebo/morse (io)*	*amoeba (g)*
atm	*awpmd/cut*	*beck (go)*
body/nparticle	*body/rounded/polygon*	*body/rounded/polyhedron*
bop	*born (go)*	*born/coul/dsf*
born/coul/dsf/cs	*born/coul/long (go)*	*born/coul/long/cs (g)*
born/coul/msm (o)	*born/coul/wolf (go)*	*born/coul/wolf/cs (g)*
born/gauss	*bpm/spring*	*brownian (o)*
brownian/poly (o)	*buck (giko)*	*buck/coul/cut (giko)*
buck/coul/long (giko)	*buck/coul/long/cs*	*buck/coul/msm (o)*
buck/long/coul/long (o)	*buck/mdf*	*buck6d/coul/gauss/dsf*
buck6d/coul/gauss/long	*colloid (go)*	*comb (o)*
comb3	*cosine/squared*	*coul/cut (gko)*
coul/cut/dielectric	*coul/cut/global (o)*	*coul/cut/soft (o)*
coul/debye (gko)	*coul/diel (o)*	*coul/dsf (gko)*
coul/exclude	*coul/long (gko)*	*coul/long/cs (g)*
coul/long/dielectric	*coul/long/soft (o)*	*coul/msm (o)*
coul/slater/cut	*coul/slater/long (g)*	*coul/shield*
coul/streitz	*coul/tt*	*coul/wolf (ko)*
coul/wolf/cs	*dpd (giko)*	*dpd/coul/slater/long (g)*
dpd/ext (ko)	*dpd/ext/tstat (ko)*	*dpd/fdt*
dpd/fdt/energy (k)	*dpd/tstat (gko)*	*dsmc*
e3b	*drip*	*eam (gikot)*
eam/alloy (gikot)	*eam/cd*	*eam/cd/old*
eam/fs (gikot)	*eam/he*	*edip (o)*
edip/multi	*edpd (g)*	*eff/cut*
eim (o)	*exp6/rx (k)*	*extep*
gauss (go)	*gauss/cut (o)*	*gayberne (gio)*
gran/hertz/history (o)	*gran/hooke (o)*	*gran/hooke/history (ko)*
granular	*gw*	*gw/zbl*
harmonic/cut (o)	*hbond/dreiding/lj (o)*	*hbond/dreiding/morse (o)*
hdnnp	*hippo (g)*	*ilp/graphene/hbn (t)*
ilp/tmd (t)	*kolmogorov/crespi/full*	*kolmogorov/crespi/z*
lcbop	*lebedeva/z*	*lennard/mdf*
lepton (o)	*lepton/coul (o)*	*lepton/sphere (o)*
line/lj	*lj/charmm/coul/charmm (giko)*	*lj/charmm/coul/charmm/implicit (ko)*
lj/charmm/coul/long (gikot)	*lj/charmm/coul/long/soft (o)*	*lj/charmm/coul/msm (o)*
lj/charmmfsw/coul/charmmfsh	*lj/charmmfsw/coul/long (k)*	*lj/class2 (gko)*
lj/class2/coul/cut (ko)	*lj/class2/coul/cut/soft*	*lj/class2/coul/long (gko)*
lj/class2/coul/long/cs	*lj/class2/coul/long/soft*	*lj/class2/soft*
lj/cubic (go)	*lj/cut (gikot)*	*lj/cut/coul/cut (gko)*
lj/cut/coul/cut/dielectric (o)	*lj/cut/coul/cut/soft (go)*	*lj/cut/coul/debye (gko)*
lj/cut/coul/debye/dielectric (o)	*lj/cut/coul/dsf (gko)*	*lj/cut/coul/long (gikot)*

continues on next page

Table 3 – continued from previous page

lj/cut/coul/long/cs	*lj/cut/coul/long/dielectric (o)*	*lj/cut/coul/long/soft (go)*
lj/cut/coul/msm (go)	*lj/cut/coul/msm/dielectric*	*lj/cut/coul/wolf (o)*
lj/cut/dipole/cut (gko)	*lj/cut/dipole/long (g)*	*lj/cut/dipole/sf (go)*
lj/cut/soft (o)	*lj/cut/sphere (o)*	*lj/cut/thole/long (o)*
lj/cut/tip4p/cut (o)	*lj/cut/tip4p/long (got)*	*lj/cut/tip4p/long/soft (o)*
lj/expand (gko)	*lj/expand/coul/long (gk)*	*lj/expand/sphere (o)*
lj/gromacs (gko)	*lj/gromacs/coul/gromacs (ko)*	*lj/long/coul/long (iot)*
lj/long/coul/long/dielectric	*lj/long/dipole/long*	*lj/long/tip4p/long (o)*
lj/mdf	*lj/relres (o)*	*lj/spica (gko)*
lj/spica/coul/long (gko)	*lj/spica/coul/msm (o)*	*lj/sf/dipole/sf (go)*
lj/smooth (go)	*lj/smooth/linear (o)*	*lj/switch3/coulgauss/long*
lj96/cut (go)	*local/density*	*lubricate (o)*
lubricate/poly (o)	*lubricateU*	*lubricateU/poly*
mdpd (g)	*mdpd/rhosum*	*meam (k)*
meam/ms (k)	*meam/spline (o)*	*meam/sw/spline*
mesocnt	*mesocnt/viscous*	*mgpt*
mie/cut (g)	*mliap (k)*	*mm3/switch3/coulgauss/long*
momb	*morse (gkot)*	*morse/smooth/linear (o)*
morse/soft	*multi/lucy*	*multi/lucy/rx (k)*
nb3b/harmonic	*nb3b/screened*	*nm/cut (o)*
nm/cut/coul/cut (o)	*nm/cut/coul/long (o)*	*nm/cut/split*
oxdna/coaxstk	*oxdna/excv*	*oxdna/hbond*
oxdna/stk	*oxdna/xstk*	*oxdna2/coaxstk*
oxdna2/dh	*oxdna2/excv*	*oxdna2/hbond*
oxdna2/stk	*oxdna2/xstk*	*oxrna2/excv*
oxrna2/hbond	*oxrna2/dh*	*oxrna2/stk*
oxrna2/xstk	*oxrna2/coaxstk*	*pace (k)*
pace/extrapolation (k)	*pedone (o)*	*pod (k)*
peri/eps	*peri/lps (o)*	*peri/pmb (o)*
peri/ves	*polymorphic*	*python*
quip	*rann*	*reaxff (ko)*
rebo (io)	*rebomos (o)*	*resquared (go)*
rheo	*rheo/solid*	*saip/metal (t)*
sdpd/taitwater/isothermal	*smatb*	*smatb/single*
smd/hertz	*smd/tlsph*	*smd/tri_surface*
smd/ulsph	*smtbq*	*snap (ik)*
soft (gko)	*sph/heatconduction (g)*	*sph/idealgas*
sph/lj (g)	*sph/rhosum*	*sph/taitwater (g)*
sph/taitwater/morris	*spin/dipole/cut*	*spin/dipole/long*
spin/dmi	*spin/exchange*	*spin/exchange/biquadratic*
spin/magelec	*spin/neel*	*srp*
srp/react	*sw (giko)*	*sw/angle/table*
sw/mod (o)	*table (gko)*	*table/rx (k)*
tdpd	*tersoff (giko)*	*tersoff/mod (gko)*
tersoff/mod/c (o)	*tersoff/table (o)*	*tersoff/zbl (gko)*
thole	*threebody/table*	*tip4p/cut (o)*
tip4p/long (o)	*tip4p/long/soft (o)*	*tri/lj*
ufm (got)	*uf3 (k)*	*vashishta (gko)*
vashishta/table (o)	*wf/cut*	*ylz*
yukawa (gko)	*yukawa/colloid (gko)*	*zbl (gko)*

5.9 Bond styles

All LAMMPS *bond_style* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

none	*zero*	*hybrid (k)*		
bpm/rotational	*bpm/spring*	*class2 (ko)*	*fene (iko)*	*fene/expand (o)*
fene/nm	*gaussian*	*gromos (o)*	*harmonic (iko)*	*harmonic/restrain*
harmonic/shift (o)	*harmonic/shift/cut (o)*	*lepton (o)*	*mesocnt*	*mm3*
morse (o)	*nonlinear (o)*	*oxdna/fene*	*oxdna2/fene*	*oxrna2/fene*
quartic (o)	*rheo/shell*	*special*	*table (o)*	

5.10 Angle styles

All LAMMPS *angle_style* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

none	*zero*	*hybrid (k)*		
amoeba	*charmm (iko)*	*class2 (ko)*	*class2/p6*	*cosine (ko)*
cosine/buck6d	*cosine/delta (o)*	*cosine/periodic (o)*	*cosine/shift (o)*	*cosine/shift/exp (o)*
cosine/squared (o)	*cosine/squared/restricted (o)*	*cross*	*dipole (o)*	*fourier (o)*
fourier/simple (o)	*gaussian*	*harmonic (iko)*	*lepton (o)*	*mesocnt*
mm3	*quartic (o)*	*spica (ko)*	*table (o)*	

5.11 Dihedral styles

All LAMMPS *dihedral_style* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

none	*zero*	*hybrid (k)*		
charmm (iko)	*charmmfsw (k)*	*class2 (ko)*	*cosine/shift/exp (o)*	*cosine/squared/restricted*
fourier (io)	*harmonic (iko)*	*helix (o)*	*lepton (o)*	*multi/harmonic (o)*
nharmonic (o)	*opls (iko)*	*quadratic (o)*	*spherical*	*table (o)*
table/cut				

5.12 Improper styles

All LAMMPS *improper_style* commands. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

none	*zero*	*hybrid (k)*		
amoeba	*class2 (ko)*	*cossq (o)*	*cvff (io)*	*distance*
distharm	*fourier (o)*	*harmonic (iko)*	*inversion/harmonic*	*ring (o)*
sqdistharm	*umbrella (o)*			

5.13 KSpace styles

All LAMMPS *kspace_style* solvers. Some styles have accelerated versions. This is indicated by additional letters in parenthesis: g = GPU, i = INTEL, k = KOKKOS, o = OPENMP, t = OPT.

ewald (o)	*ewald/disp*	*ewald/disp/dipole*	*ewald/dipole*	*ewald/dipole/spin*
ewald/electrode	*msm (o)*	*msm/cg (o)*	*msm/dielectric*	*pppm (giko)*
pppm/cg (o)	*pppm/dipole*	*pppm/dipole/spin*	*pppm/dielectric*	*pppm/disp (io)*
pppm/disp/tip4p (o)	*pppm/disp/dielectric*	*pppm/stagger*	*pppm/tip4p (o)*	*pppm/dielectric*
pppm/electrode (i)	*scafacos*			

5.14 Dump styles

An alphabetic list of all LAMMPS *dump* commands.

atom	*atom/adios*	*atom/gz*	*atom/zstd*	*cfg*	*cfg/gz*
cfg/uef	*cfg/zstd*	*custom*	*custom/adios*	*custom/gz*	*custom/zstd*
dcd	*grid*	*grid/vtk*	*h5md*	*image*	*local*
local/gz	*local/zstd*	*molfile*	*movie*	*netcdf*	*netcdf/mpiio*
vtk	*xtc*	*xyz*	*xyz/gz*	*xyz/zstd*	*yaml*

5.15 Removed commands and packages

This page lists LAMMPS commands and packages that have been removed from the distribution and provides suggestions for alternatives or replacements. LAMMPS has special dummy styles implemented, that will stop LAMMPS and print a suitable error message in most cases, when a style/command is used that has been removed or will replace the command with the direct alternative (if available) and print a warning.

5.15.1 restart2data tool

Changed in version 23Nov2013.

The functionality of the restart2data tool has been folded into the LAMMPS executable directly instead of having a separate tool. A combination of the commands *read_restart* and *write_data* can be used to the same effect. For added convenience this conversion can also be triggered by *command line flags*

5.15.2 Fix ave/spatial and fix ave/spatial/sphere

Deprecated since version 11Dec2015.

The fixes ave/spatial and ave/spatial/sphere have been removed from LAMMPS since they were superseded by the more general and extensible "chunk infrastructure". Here the system is partitioned in one of many possible ways through the *compute chunk/atom* command and then averaging is done using *fix ave/chunk*. Please refer to the *chunk HOWTO* section for an overview.

5.15.3 Box command

Deprecated since version 22Dec2022.

The *box* command has been removed and the LAMMPS code changed so it won't be needed. If present, LAMMPS will ignore the command and print a warning.

5.15.4 Reset_ids, reset_atom_ids, reset_mol_ids commands

Deprecated since version 22Dec2022.

The *reset_ids*, *reset_atom_ids*, and *reset_mol_ids* commands have been folded into the *reset_atoms* command. If present, LAMMPS will replace the commands accordingly and print a warning.

5.15.5 LATTE package

Deprecated since version 15Jun2023.

The LATTE package with the fix latte command was removed from LAMMPS. This functionality has been superseded by *fix mdi/qm* and *fix mdi/qmmm* from the *MDI package*. These fixes are compatible with several quantum software packages, including LATTE. See the examples/QUANTUM dir and the *MDI coupling HOWTO* page. MDI supports running LAMMPS with LATTE as a plugin library (similar to the way fix latte worked), as well as on a different set of MPI processors.

5.15.6 MEAM package

The MEAM package in Fortran has been replaced by a C++ implementation. The code in the *MEAM package* is a translation of the Fortran code of MEAM into C++, which removes several restrictions (e.g. there can be multiple instances in hybrid pair styles) and allows for some optimizations leading to better performance. The pair style *meam* has the exact same syntax. For a transition period the C++ version of MEAM was called USER-MEAMC so it could coexist with the Fortran version.

5.15.7 Minimize style fire/old

Deprecated since version 8Feb2023.

Minimize style *fire/old* has been removed. Its functionality can be reproduced with *fire* with specific options. Please see the *min_modify command* documentation for details.

5.15.8 Pair style mesont/tpm, compute style mesont, atom style mesont

Deprecated since version 8Feb2023.

Pair style *mesont/tpm*, compute style *mesont*, and atom style *mesont* have been removed from the *MESONT package*. The same functionality is available through *pair style mesocnt*, *bond style mesocnt* and *angle style mesocnt*.

5.15.9 MPIIO package

Deprecated since version 21Nov2023.

The MPIIO package has been removed from LAMMPS since it was unmaintained for many years and thus not updated to incorporate required changes that had been applied to the corresponding non-MPIIO commands. As a consequence the MPIIO commands had become unreliable and sometimes crashing LAMMPS or corrupting data. Similar functionality is available through the *ADIOS package* and the *NETCDF package*. Also, the *dump_modify nfile or dump_modify fileper* keywords may be used for an efficient way of writing out dump files when running on large numbers of processors. Similarly, the "nfile" and "fileper" keywords exist for restarts: see *restart*, *read_restart*, *write_restart*.

5.15.10 MSCG package

Deprecated since version 21Nov2023.

The MSCG package has been removed from LAMMPS since it was unmaintained for many years and instead superseded by the OpenMSCG software of the Voth group at the University of Chicago, which can be used independent from LAMMPS.

5.15.11 REAX package

The REAX package has been removed since it was superseded by the *REAXFF package*. The REAXFF package has been tested to yield equivalent results to the REAX package, offers better performance, supports OpenMP multi-threading via OPENMP, and GPU and threading parallelization through KOKKOS. The new pair styles are not syntax compatible with the removed reax pair style, so input files will have to be adapted. The REAXFF package was originally called USER-REAXC.

5.15.12 USER-REAXC package

Deprecated since version 7Feb2024.

The USER-REAXC package has been renamed to *REAXFF*. In the process also the pair style and related fixes were renamed to use the "reaxff" string instead of "reax/c". For a while LAMMPS was maintaining backward compatibility by providing aliases for the styles. These have been removed, so using "reaxff" is now *required*.

5.15.13 USER-CUDA package

The USER-CUDA package had been removed, since it had been unmaintained for a long time and had known bugs and problems. Significant parts of the design were transferred to the *KOKKOS package*, which has similar performance characteristics on NVIDIA GPUs. Both, the KOKKOS and the *GPU package* are maintained and allow running LAMMPS with GPU acceleration.

5.15.14 i-PI tool

Changed in version 27Jun2024.

The i-PI tool has been removed from the LAMMPS distribution. Instead, instructions to install i-PI from PyPI via pip are provided.

5.15.15 LAMMPS shell

Changed in version 29Aug2024.

The LAMMPS shell has been removed from the LAMMPS distribution. Users are encouraged to use the *LAMMPS-GUI* tool instead.