kspace_style command

Syntax:

```
kspace_style style value
```

- style = *none* or *ewald* or *ewald/disp* or *ewald/omp* or *pppm* or *pppm/cg* or *pppm/disp* or *pppm/tip4p* or *pppm/stagger* or *pppm/disp/tip4p* or *pppm/gpu* or *pppm/kk* or *pppm/omp* or *pppm/cg/omp* or *pppm/tip4p/omp* or *msm* or *msm/cg* or *msm/omp* or *msm/cg/omp*

```
none value = none
  ewald value = accuracy
    accuracy = desired relative error in forces
  ewald/disp value = accuracy
    accuracy = desired relative error in forces
  ewald/omp value = accuracy
    accuracy = desired relative error in forces
  pppm value = accuracy (fscale)
    accuracy = desired relative error in forces
    fscale = scale factor in forces (optional)
  pppm/cg value = accuracy (smallq)
    accuracy = desired relative error in forces
    smallq = cutoff for charges to be considered (optional) (charge units)
  pppm/disp value = accuracy
    accuracy = desired relative error in forces
  pppm/tip4p value = accuracy (fscale)
    accuracy = desired relative error in forces
    fscale = scale factor in forces (optional)
  pppm/disp/tip4p value = accuracy
    accuracy = desired relative error in forces
  pppm/gpu value = accuracy
    accuracy = desired relative error in forces
  pppm/intel value = accuracy
    accuracy = desired relative error in forces
  pppm/kk value = accuracy
    accuracy = desired relative error in forces
  pppm/omp value = accuracy
    accuracy = desired relative error in forces
  pppm/cg/omp value = accuracy
    accuracy = desired relative error in forces
  pppm/disp/intel value = accuracy
    accuracy = desired relative error in forces
  pppm/tip4p/omp value = accuracy
    accuracy = desired relative error in forces
  pppm/stagger value = accuracy
    accuracy = desired relative error in forces
  msm value = accuracy
    accuracy = desired relative error in forces
  msm/cg value = accuracy (smallq)
    accuracy = desired relative error in forces
    smallq = cutoff for charges to be considered (optional) (charge units)
  msm/omp value = accuracy
    accuracy = desired relative error in forces
  msm/cg/omp value = accuracy (smallq)
    accuracy = desired relative error in forces
```

```
   smallq = cutoff for charges to be considered (optional) (charge units)
```

Examples:

```
kspace_style pppm 1.0e-4
kspace_style pppm/cg 1.0e-5 1.0e-6
kspace_style msm 1.0e-4
kspace_style pppm/tip4p 1.0e-6 0
kspace_style none
```

Description:

Define a long-range solver for LAMMPS to use each timestep to compute long-range Coulombic interactions or long-range 1/r^6 interactions. Most of the long-range solvers perform their computation in K-space, hence the name of this command.

When such a solver is used in conjunction with an appropriate pair style, the cutoff for Coulombic or 1/r^N interactions is effectively infinite. If the Coulombic case, this means each charge in the system interacts with charges in an infinite array of periodic images of the simulation domain.

Note that using a long-range solver requires use of a matching pair style to perform consistent short-range pairwise calculations. This means that the name of the pair style contains a matching keyword to the name of the KSpace style, as in this table:

Pair style	KSpace style
coul/long	ewald or pppm
coul/msm	msm
lj/long or buck/long	disp (for dispersion)
tip4p/long	tip4p

The *ewald* style performs a standard Ewald summation as described in any solid-state physics text.

The *ewald/disp* style adds a long-range dispersion sum option for 1/r^6 potentials and is useful for simulation of interfaces (Veld). It also performs standard Coulombic Ewald summations, but in a more efficient manner than the *ewald* style. The 1/r^6 capability means that Lennard-Jones or Buckingham potentials can be used without a cutoff, i.e. they become full long-range potentials. The *ewald/disp* style can also be used with point-dipoles (Toukmaji) and is currently the only kspace solver in LAMMPS with this capability.

The *pppm* style invokes a particle-particle particle-mesh solver (Hockney) which maps atom charge to a 3d mesh, uses 3d FFTs to solve Poisson's equation on the mesh, then interpolates electric fields on the mesh points back to the atoms. It is closely related to the particle-mesh Ewald technique (PME) (Darden) used in AMBER and CHARMM. The cost of traditional Ewald summation scales as N^(3/2) where N is the number of atoms in the system. The PPPM solver scales as Nlog(N) due to the FFTs, so it is almost always a faster choice (Pollock).

The *pppm/cg* style is identical to the *pppm* style except that it has an optimization for systems where most particles are uncharged. Similarly the *msm/cg* style implements the same optimization for *msm*. The optional *smallq* argument defines the cutoff for the absolute charge value which determines whether a particle is considered charged or not. Its default value is 1.0e-5.

The *pppm/tip4p* style is identical to the *pppm* style except that it adds a charge at the massless 4th site in each TIP4P water molecule. It should be used with pair styles with a *tip4p/long* in their style name.

The *pppm/stagger* style performs calculations using two different meshes, one shifted slightly with respect to the other. This can reduce force aliasing errors and increase the accuracy of the method for a given mesh size. Or a coarser mesh can be used for the same target accuracy, which saves CPU time. However, there is a trade-off since FFTs on two meshes are now performed which increases the computation required. See (Cerutti), (Neelov), and (Hockney) for details of the method.

For high relative accuracy, using staggered PPPM allows the mesh size to be reduced by a factor of 2 in each dimension as compared to regular PPPM (for the same target accuracy). This can give up to a 4x speedup in the KSpace time (8x less mesh points, 2x more expensive). However, for low relative accuracy, the staggered PPPM mesh size may be essentially the same as for regular PPPM, which means the method will be up to 2x slower in the KSpace time (simply 2x more expensive). For more details and timings, see the Speed tips doc page.

NOTE: Using *pppm/stagger* may not give the same increase in the accuracy of energy and pressure as it does in forces, so some caution must be used if energy and/or pressure are quantities of interest, such as when using a barostat.

The *pppm/disp* and *pppm/disp/tip4p* styles add a mesh-based long-range dispersion sum option for 1/r^6 potentials (Isele-Holder), similar to the *ewald/disp* style. The 1/r^6 capability means that Lennard-Jones or Buckingham potentials can be used without a cutoff, i.e. they become full long-range potentials.

For these styles, you will possibly want to adjust the default choice of parameters by using the kspace_modify command. This can be done by either choosing the Ewald and grid parameters, or by specifying separate accuracies for the real and kspace calculations. When not making any settings, the simulation will stop with an error message. Further information on the influence of the parameters and how to choose them is described in (Isele-Holder), (Isele-Holder2) and the Howto dispersion doc page.

NOTE: All of the PPPM styles can be used with single-precision FFTs by using the compiler switch -DFFT_SINGLE for the FFT_INC setting in your lo-level Makefile. This setting also changes some of the PPPM operations (e.g. mapping charge to mesh and interpolating electric fields to particles) to be performed in single precision. This option can speed-up long-range calculations, particularly in parallel or on GPUs. The use of the -DFFT_SINGLE flag is discussed on the Build settings doc page. MSM does not currently support the -DFFT_SINGLE compiler switch.

The *msm* style invokes a multi-level summation method MSM solver, (Hardy) or (Hardy2), which maps atom charge to a 3d mesh, and uses a multi-level hierarchy of coarser and coarser meshes on which direct coulomb solves are done. This method does not use FFTs and scales as N. It may therefore be faster than the other K-space solvers for relatively large problems when running on large core counts. MSM can also be used for non-periodic boundary conditions and for mixed periodic and non-periodic boundaries.

MSM is most competitive versus Ewald and PPPM when only relatively low accuracy forces, about 1e-4 relative error or less accurate, are needed. Note that use of a larger coulomb cutoff (i.e. 15 angstroms instead of 10 angstroms) provides better MSM accuracy for both the real space and grid computed forces.

Currently calculation of the full pressure tensor in MSM is expensive. Using the kspace_modify *pressure/scalar yes* command provides a less expensive way to compute the scalar pressure (Pxx + Pyy + Pzz)/3.0. The scalar pressure can be used, for example, to run an isotropic barostat. If the full pressure tensor is needed, then calculating the pressure at every timestep or using a fixed pressure simulation with MSM will cause the code to run slower.

The specified *accuracy* determines the relative RMS error in per-atom forces calculated by the long-range solver. It is set as a dimensionless number, relative to the force that two unit point charges (e.g. 2 monovalent ions) exert on each other at a distance of 1 Angstrom. This reference value was chosen as representative of the magnitude of electrostatic forces in atomic systems. Thus an accuracy value of 1.0e-4 means that the RMS error will be a factor of 10000 smaller than the reference force.

The accuracy setting is used in conjunction with the pairwise cutoff to determine the number of K-space vectors for style *ewald* or the grid size for style *pppm* or *msm*.

Note that style *pppm* only computes the grid size at the beginning of a simulation, so if the length or triclinic tilt of the simulation cell increases dramatically during the course of the simulation, the accuracy of the simulation may degrade. Likewise, if the kspace_modify slab option is used with shrink-wrap boundaries in the z-dimension, and the box size changes dramatically in z. For example, for a triclinic system with all three tilt factors set to the maximum limit, the PPPM grid should be increased roughly by a factor of 1.5 in the y direction and 2.0 in the z direction as compared to the same system using a cubic orthogonal simulation cell. One way to handle this issue if you have a long simulation where the box size changes dramatically, is to break it into shorter simulations (multiple run commands). This works because the grid size is re-computed at the beginning of each run. Another way to ensure the described accuracy requirement is met is to run a short simulation at the maximum expected tilt or length, note the required grid size, and then use the kspace_modify *mesh* command to manually set the PPPM grid size to this value for the long run. The simulation then will be "too accurate" for some portion of the run.

RMS force errors in real space for *ewald* and *pppm* are estimated using equation 18 of (Kolafa), which is also referenced as equation 9 of (Petersen). RMS force errors in K-space for *ewald* are estimated using equation 11 of (Petersen), which is similar to equation 32 of (Kolafa). RMS force errors in K-space for *pppm* are estimated using equation 38 of (Deserno). RMS force errors for *msm* are estimated using ideas from chapter 3 of (Hardy), with equation 3.197 of particular note. When using *msm* with non-periodic boundary conditions, it is expected that the error estimation will be too pessimistic. RMS force errors for dipoles when using *ewald/disp* are estimated using equations 33 and 46 of (Wang).

See the kspace_modify command for additional options of the K-space solvers that can be set, including a *force* option for setting an absolute RMS error in forces, as opposed to a relative RMS error.

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

More specifically, the *pppm/gpu* style performs charge assignment and force interpolation calculations on the GPU. These processes are performed either in single or double precision, depending on whether the -DFFT_SINGLE setting was specified in your lo-level Makefile, as discussed above. The FFTs themselves are still calculated on the CPU. If *pppm/gpu* is used with a GPU-enabled pair style, part of the PPPM calculation can be performed concurrently on the GPU while other calculations for non-bonded and bonded force calculation are performed on the CPU.

The *pppm/kk* style also performs charge assignment and force interpolation calculations on the GPU while the FFTs themselves are calculated on the CPU in non-threaded mode.

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

See the Speed packages doc page for more instructions on how to use the accelerated styles effectively.

Restrictions:

Note that the long-range electrostatic solvers in LAMMPS assume conducting metal (tinfoil) boundary conditions for both charge and dipole interactions. Vacuum boundary conditions are not currently supported.

The *ewald/disp*, *ewald*, *pppm*, and *msm* styles support non-orthogonal (triclinic symmetry) simulation boxes. However, triclinic simulation cells may not yet be supported by suffix versions of these styles.

All of the kspace styles are part of the KSPACE package. They are only enabled if LAMMPS was built with that package. See the Build package doc page for more info. Note that the KSPACE package is installed by default.

For MSM, a simulation must be 3d and one can use any combination of periodic, non-periodic, or shrink-wrapped boundaries (specified using the boundary command).

For Ewald and PPPM, a simulation must be 3d and periodic in all dimensions. The only exception is if the slab option is set with kspace_modify, in which case the xy dimensions must be periodic and the z dimension must be non-periodic.

Related commands:

kspace_modify, pair_style lj/cut/coul/long, pair_style lj/charmm/coul/long, pair_style lj/long/coul/long, pair_style buck/coul/long

Default:

```
kspace_style none
```

(Darden) Darden, York, Pedersen, J Chem Phys, 98, 10089 (1993).

(Deserno) Deserno and Holm, J Chem Phys, 109, 7694 (1998).

(Hockney) Hockney and Eastwood, Computer Simulation Using Particles, Adam Hilger, NY (1989).

(Kolafa) Kolafa and Perram, Molecular Simulation, 9, 351 (1992).

(Petersen) Petersen, J Chem Phys, 103, 3668 (1995).

(Wang) Wang and Holm, J Chem Phys, 115, 6277 (2001).

(Pollock) Pollock and Glosli, Comp Phys Comm, 95, 93 (1996).

(Cerutti) Cerutti, Duke, Darden, Lybrand, Journal of Chemical Theory and Computation 5, 2322 (2009)

(Neelov) Neelov, Holm, J Chem Phys 132, 234103 (2010)

(Veld) In 't Veld, Ismail, Grest, J Chem Phys, 127, 144711 (2007).

(Toukmaji) Toukmaji, Sagui, Board, and Darden, J Chem Phys, 113, 10913 (2000).

(Isele-Holder) Isele-Holder, Mitchell, Ismail, J Chem Phys, 137, 174107 (2012).

(Isele-Holder2) Isele-Holder, Mitchell, Hammond, Kohlmeyer, Ismail, J Chem Theory Comput 9, 5412 (2013).

(Hardy) David Hardy thesis: Multilevel Summation for the Fast Evaluation of Forces for the Simulation of Biomolecules, University of Illinois at Urbana-Champaign, (2006).

(Hardy2) Hardy, Stone, Schulten, Parallel Computing 35 (2009) 164-177.