General information

Default values are given in parentheses, or listed first among choices. The first option in the list is always the default option. Units are given in square brackets. The difference between a dash and an underscore is ignored.

A sample mdp file is available. This should be appropriate to start a normal simulation. Edit it to suit your specific needs and desires.

Preprocessing

include

directories to include in your topology. Format: -I/home/john/mylib -I../otherlib

define

defines to pass to the preprocessor, default is no defines. You can use any defines to control options in your customized topology files. Options that act on existing top file mechanisms include

-DFLEXIBLE will use flexible water instead of rigid water into your topology, this can be useful for normal mode analysis.

-DPOSRES will trigger the inclusion of posre.itp into your topology, used for implementing position restraints.

Run control

integrator

(Despite the name, this list includes algorithms that are not actually integrators over time. integrator=steep and all entries following it are in this category)

md

A leap-frog algorithm for integrating Newton's equations of motion.

md-vv

A velocity Verlet algorithm for integrating Newton's equations of motion. For constant NVE simulations started from corresponding points in the same trajectory, the trajectories are analytically, but not binary, identical to the integrator=md leap-frog integrator. The the kinetic energy, which is determined from the whole step velocities and is therefore slightly too high. The advantage of this integrator is more accurate, reversible Nose-Hoover and Parrinello-Rahman coupling integration based on Trotter expansion, as well as (slightly too small) full step velocity output. This all comes at the cost off extra computation, especially with constraints and extra communication in parallel. Note that for nearly all production simulations the integrator=md integrator is accurate enough.

md-vv-avek

A velocity Verlet algorithm identical to integrator=md-vv, except that the kinetic energy is determined as the average of the two half step kinetic energies as in the integrator=md integrator, and this thus more accurate. With Nose-Hoover and/or Parrinello-Rahman coupling this comes with a slight increase in computational cost.

sd

An accurate and efficient leap-frog stochastic dynamics integrator. With constraints, coordinates needs to be constrained twice per integration step. Depending on the computational cost of the force calculation, this can take a significant part of the simulation time. The temperature for one or more groups of atoms (tc-grps) is set with ref-t, the inverse friction constant for each group is set with tau-t. The parameter tcoupl is ignored. The random generator is initialized with ld-seed. When used as a thermostat, an appropriate value for tau-t is 2 ps, since this results in a friction that is lower than the internal friction of water, while it is high enough to remove excess heat NOTE: temperature deviations decay twice as fast as with a Berendsen thermostat with the same tau-t.

bd

An Euler integrator for Brownian or position Langevin dynamics, the velocity is the force divided by a friction coefficient (bd-fric) plus random thermal noise (ref-t). When bd-fric is 0, the friction coefficient for each particle is calculated as mass/ tau-t, as for the integrator integrator=sd. The random generator is initialized with ld-seed.

steep

A steepest descent algorithm for energy minimization. The maximum step size is emstep, the tolerance is emtol.

cg

A conjugate gradient algorithm for energy minimization, the tolerance is emtol. CG is more efficient when a steepest descent step is done every once in a while, this is determined by nstcgsteep. For a minimization prior to a normal mode analysis, which requires a very high accuracy, GROMACS should be compiled in double precision.

l-bfgs

A quasi-Newtonian algorithm for energy minimization according to the low-memory Broyden-Fletcher-Goldfarb-Shanno approach. In practice this seems to converge faster than Conjugate Gradients, but due to the correction steps necessary it is not (yet) parallelized.

nm

Normal mode analysis is performed on the structure in the tpr file. GROMACS should be compiled in double precision.

tpi

Test particle insertion. The last molecule in the topology is the test particle. A trajectory must be provided to mdrun -rerun. This trajectory should not contain the molecule to be inserted. Insertions are performed nsteps times in each frame at random locations and with random orientiations of the molecule. When nstlist is larger than one, nstlist insertions are performed in a sphere with radius rtpi around a the same random location using the same pair list. Since pair list construction is expensive, one can perform several extra insertions with the same list almost for free. The random seed is set with ld-seed. The temperature for the Boltzmann weighting is set with ref-t, this should match the temperature of the simulation of the original trajectory. Dispersion correction is implemented correctly for TPI. All relevant quantities are written to the file specified with mdrun -tpi. The distribution of insertion energies is written to the file specified with mdrun -tpid. No trajectory or energy file is written. Parallel TPI gives identical results to single-node TPI. For charged molecules, using PME with a fine grid is most accurate and also efficient, since the potential in the system only needs to be calculated once per frame.

tpic

Test particle insertion into a predefined cavity location. The procedure is the same as for integrator=tpi, except that one coordinate extra is read from the trajectory, which is used as the insertion location. The molecule to be inserted should be centered at 0,0,0. GROMACS does not do this for you, since for different situations a different way of centering might be optimal. Also rtpi sets the radius for the sphere around this location. Neighbor searching is done only once per frame, nstlist is not used. Parallel integrator=tpic gives identical results to single-rank integrator=tpic.

tinit

(0) [ps] starting time for your run (only makes sense for time-based integrators)

dt

(0.001) [ps] time step for integration (only makes sense for time-based integrators)

nsteps

(0) maximum number of steps to integrate or minimize, -1 is no maximum

init-step

(0) The starting step. The time at an step i in a run is calculated as: t = tinit + dt * (init-step + i). The free-energy lambda is calculated as: lambda = init-lambda + delta-lambda * (init-step + i). Also non-equilibrium MD parameters can depend on the step number. Thus for exact restarts or redoing part of a run it might be necessary to set init-step to the step number of the restart frame. gmx convert-tpr does this automatically.

simulation-part

(0) A simulation can consist of multiple parts, each of which has a part number. This option specifies what that number will be, which helps keep track of parts that are logically the same simulation. This option is generally useful to set only when coping with a crashed simulation where files were lost.

comm-mode

Linear

Remove center of mass translational velocity

Angular

Remove center of mass translational and rotational velocity around the center of mass

Linear-acceleration-correction

Remove center of mass translational velocity. Correct the center of mass position assuming linear acceleration over nstcomm steps. This is useful for cases where an acceleration is expected on the center of mass which is nearly constant over mdp:*nstcomm* steps. This can occur for example when pulling on a group using an absolute reference.

None

No restriction on the center of mass motion

nstcomm

(100) [steps] frequency for center of mass motion removal

comm-grps

group(s) for center of mass motion removal, default is the whole system

Langevin dynamics

bd-fric

(0) [amu ps-1] Brownian dynamics friction coefficient. When bd-fric is 0, the friction coefficient for each particle is calculated as mass/ tau-t.

ld-seed

(-1) [integer] used to initialize random generator for thermal noise for stochastic and Brownian dynamics. When ld-seed is set to -1, a pseudo random seed is used. When running BD or SD on multiple processors, each processor uses a seed equal to ld-seed plus the processor number.

Energy minimization

emtol

(10.0) [kJ mol-1 nm-1] the minimization is converged when the maximum force is smaller than this value

emstep

(0.01) [nm] initial step-size

nstcgsteep

(1000) [steps] frequency of performing 1 steepest descent step while doing conjugate gradient energy minimization.

nbfgscorr

(10) Number of correction steps to use for L-BFGS minimization. A higher number is (at least theoretically) more accurate, but slower.

Shell Molecular Dynamics

When shells or flexible constraints are present in the system the positions of the shells and the lengths of the flexible constraints are optimized at every time step until either the RMS force on the shells and constraints is less than emtol, or a maximum number of iterations niter has been reached. Minimization is converged when the maximum force is smaller than emtol. For shell MD this value should be 1.0 at most.

niter

(20) maximum number of iterations for optimizing the shell positions and the flexible constraints.

fcstep

(0) [ps^2] the step size for optimizing the flexible constraints. Should be chosen as mu/(d2V/dq2) where mu is the reduced mass of two particles in a flexible constraint and d2V/dq2 is the second derivative of the potential in the constraint direction. Hopefully this number does not differ too much between the flexible constraints, as the number of iterations and thus the runtime is very sensitive to fcstep. Try several values!

Test particle insertion

rtpi

(0.05) [nm] the test particle insertion radius, see integrators integrator=tpi and integrator=tpic

Output control

nstxout

(0) [steps] number of steps that elapse between writing coordinates to output trajectory file, the last coordinates are always written

nstvout

(0) [steps] number of steps that elapse between writing velocities to output trajectory, the last velocities are always written

nstfout

(0) [steps] number of steps that elapse between writing forces to output trajectory.

nstlog

(1000) [steps] number of steps that elapse between writing energies to the log file, the last energies are always written

nstcalcenergy

(100) number of steps that elapse between calculating the energies, 0 is never. This option is only relevant with dynamics. This option affects the performance in parallel simulations, because calculating energies requires global communication between all processes which can become a bottleneck at high parallelization.

nstenergy

(1000) [steps] number of steps that else between writing energies to energy file, the last energies are always written, should be a multiple of nstcalcenergy. Note that the exact sums and fluctuations over all MD steps modulo nstcalcenergy are stored in the energy file, so gmx energy can report exact energy averages and fluctuations also when nstenergy > 1

nstxout-compressed

(0) [steps] number of steps that elapse between writing position coordinates using lossy compression

compressed-x-precision

(1000) [real] precision with which to write to the compressed trajectory file

compressed-x-grps

group(s) to write to the compressed trajectory file, by default the whole system is written (if nstxout-compressed > 0)

energygrps

group(s) for which to write to write short-ranged non-bonded potential energies to the energy file (not supported on GPUs)

Neighbor searching

cutoff-scheme

Verlet

Generate a pair list with buffering. The buffer size is automatically set based on verlet-buffer-tolerance, unless this is set to -1, in which case rlist will be used. This option has an explicit, exact cut-off at rvdw equal to rcoulomb, unless PME or Ewald is used, in which case rcoulomb > rvdw is allowed. Currently only cut-off, reaction-field, PME or Ewald electrostatics and plain LJ are supported. Some gmx mdrun functionality is not yet supported with the cutoff-scheme=Verlet scheme, but gmx grompp checks for this. Native GPU acceleration is only supported with cutoff-scheme=Verlet. With GPU-accelerated PME or with separate PME ranks, gmx mdrun will automatically tune the CPU/GPU load balance by scaling rcoulomb and the grid spacing. This can be turned off with mdrun -notunepme. cutoff-scheme=Verlet is faster than cutoff-scheme=group when there is no water, or if cutoff-scheme=group would use a pair-list buffer to conserve energy.

group

Generate a pair list for groups of atoms. These groups correspond to the charge groups in the topology. This was the only cut-off treatment scheme before version 4.6, and is **deprecated since 5.1**. There is no explicit buffering of the pair list. This enables efficient force calculations for water, but energy is only conserved when a buffer is explicitly added.

nstlist

(10) [steps]

>0

Frequency to update the neighbor list. When this is 0, the neighbor list is made only once. With energy minimization the pair list will be updated for every energy evaluation when nstlist is greater than 0. With cutoff-scheme=Verlet and verlet-buffer-tolerance set, nstlist is actually a minimum value and gmx mdrun might increase it, unless it is set to 1. With parallel simulations and/or non-bonded force calculation on the GPU, a value of 20 or 40 often gives the best performance. With cutoff-scheme=group and non-exact cut-off's, nstlist will affect the accuracy of your simulation and it can not be chosen freely.

0

The neighbor list is only constructed once and never updated. This is mainly useful for vacuum simulations in which all particles see each other.

<0

Unused.

ns-type

grid

Make a grid in the box and only check atoms in neighboring grid cells when constructing a new neighbor list every nstlist steps. In large systems grid search is much faster than simple search.

simple

Check every atom in the box when constructing a new neighbor list every nstlist steps (only with cutoff-scheme=group cut-off scheme).

pbc

xyz

Use periodic boundary conditions in all directions.

no

Use no periodic boundary conditions, ignore the box. To simulate without cut-offs, set all cut-offs and nstlist to 0. For best performance without cut-offs on a single MPI rank, set nstlist to zero and ns-type =simple.

xy

Use periodic boundary conditions in x and y directions only. This works only with ns-type =grid and can be used in combination with walls. Without walls or with only one wall the system size is infinite in the z direction. Therefore pressure coupling or Ewald summation methods can not be used. These disadvantages do not apply when two walls are used.

periodic-molecules

no

molecules are finite, fast molecular PBC can be used

yes

for systems with molecules that couple to themselves through the periodic boundary conditions, this requires a slower PBC algorithm and molecules are not made whole in the output

verlet-buffer-tolerance

(0.005) [kJ/mol/ps]

Useful only with the cutoff-scheme=Verlet cutoff-scheme. This sets the maximum allowed error for pair interactions per particle caused by the Verlet buffer, which indirectly sets rlist. As both nstlist and the Verlet buffer size are fixed (for performance reasons), particle pairs not in the pair list can occasionally get within the cut-off distance during nstlist -1 steps. This causes very small jumps in the energy. In a constant-temperature ensemble, these very small energy jumps can be estimated for a given cut-off and rlist. The estimate assumes a homogeneous particle distribution, hence the errors might be slightly underestimated for multi-phase systems. (See the reference manual for details). For longer pair-list life-time (nstlist -1) * dt the buffer is overestimated, because the interactions between particles are ignored. Combined with cancellation of errors, the actual drift of the total energy is usually one to two orders of magnitude smaller. Note that the generated buffer size takes into account that the GROMACS pair-list setup leads to a reduction in the drift by a factor 10, compared to a simple particle-pair based list. Without dynamics (energy minimization etc.), the buffer is 5% of the cut-off. For NVE simulations the initial temperature is used, unless this is zero, in which case a buffer of 10% is used. For NVE simulations the tolerance usually needs to be lowered to achieve proper energy conservation on the nanosecond time scale. To override the automated buffer setting, use verlet-buffer-tolerance =-1 and set rlist manually.

rlist

(1) [nm] Cut-off distance for the short-range neighbor list. With the cutoff-scheme=Verlet cutoff-scheme, this is by default set by the verlet-buffer-tolerance option and the value of rlist is ignored.

Electrostatics

coulombtype

Cut-off

Plain cut-off with pair list radius rlist and Coulomb cut-off rcoulomb, where rlist >= rcoulomb.

Ewald

Classical Ewald sum electrostatics. The real-space cut-off rcoulomb should be equal to rlist. Use *e.g.* rlist =0.9, rcoulomb =0.9. The highest magnitude of wave vectors used in reciprocal space is controlled by fourierspacing. The relative accuracy of direct/reciprocal space is controlled by ewald-rtol.

NOTE: Ewald scales as O(N^3/2) and is thus extremely slow for large systems. It is included mainly for reference - in most cases PME will perform much better.

PME

Fast smooth Particle-Mesh Ewald (SPME) electrostatics. Direct space is similar to the Ewald sum, while the reciprocal part is performed with FFTs. Grid dimensions are controlled with fourierspacing and the interpolation order with pme-order. With a grid spacing of 0.1 nm and cubic interpolation the electrostatic forces have an accuracy of 2-3*10^-4. Since the error from the vdw-cutoff is larger than this you might try 0.15 nm. When running in parallel the interpolation parallelizes better than the FFT, so try decreasing grid dimensions while increasing interpolation.

P3M-AD

Particle-Particle Particle-Mesh algorithm with analytical derivative for for long range electrostatic interactions. The method and code is identical to SPME, except that the influence function is optimized for the grid. This gives a slight increase in accuracy.

Reaction-Field

Reaction field electrostatics with Coulomb cut-off rcoulomb, where rlist >= rvdw. The dielectric constant beyond the cut-off is epsilon-rf. The dielectric constant can be set to infinity by setting epsilon-rf =0.

Generalized-Reaction-Field

Generalized reaction field with Coulomb cut-off rcoulomb, where rlist >= rcoulomb. The dielectric constant beyond the cut-off is epsilon-rf. The ionic strength is computed from the number of charged (*i.e.* with non zero charge) charge groups. The temperature for the GRF potential is set with ref-t.

Reaction-Field-zero

In GROMACS, normal reaction-field electrostatics with cutoff-scheme = cutoff-scheme=group leads to bad energy conservation. coulombtype=Reaction-Field-zero solves this by making the potential zero beyond the cut-off. It can only be used with an infinite dielectric constant (epsilon-rf =0), because only for that value the force vanishes at the cut-off. rlist should be 0.1 to 0.3 nm larger than rcoulomb to accommodate for the size of charge groups and diffusion between neighbor list updates. This, and the fact that table lookups are used instead of analytical functions make coulombtype=Reaction-Field-zero computationally more expensive than normal reaction-field.

Shift

Analogous to vdwtype=Shift for vdwtype. You might want to use coulombtype=Reaction-Field-zero instead, which has a similar potential shape, but has a physical interpretation and has better energies due to the exclusion correction terms.

Encad-Shift

The Coulomb potential is decreased over the whole range, using the definition from the Encad simulation package.

Switch

Analogous to vdwtype=Switch for vdwtype. Switching the Coulomb potential can lead to serious artifacts, advice: use coulombtype=Reaction-Field-zero instead.

User

gmx mdrun will now expect to find a file table.xvg with user-defined potential functions for repulsion, dispersion and Coulomb. When pair interactions are present, gmx mdrun also expects to find a file tablep.xvg for the pair interactions. When the same interactions should be used for non-bonded and pair interactions the user can specify the same file name for both table files. These files should contain 7 columns: the x value, f(x), -f'(x), g(x), -g'(x), h(x), -h'(x), where f(x) is the Coulomb function, g(x) the dispersion function and h(x) the repulsion function. When vdwtype is not set to User the values for g, -g', h and -h' are ignored. For the non-bonded interactions x values should run from 0 to the largest cut-off distance + table-extension and should be uniformly spaced. For the pair interactions the table length in the file will be used. The optimal spacing, which is used for non-user tables, is 0.002 nm when you run in mixed precision or 0.0005 nm when you run in double precision. The function value at x=0 is not important. More information is in the printed manual.

PME-Switch

A combination of PME and a switch function for the direct-space part (see above). rcoulomb is allowed to be smaller than rlist. This is mainly useful constant energy simulations (note that using PME with cutoff-scheme = cutoff-scheme=Verlet will be more efficient).

PME-User

A combination of PME and user tables (see above). rcoulomb is allowed to be smaller than rlist. The PME mesh contribution is subtracted from the user table by gmx mdrun. Because of this subtraction the user tables should contain about 10 decimal places.

PME-User-Switch

A combination of PME-User and a switching function (see above). The switching function is applied to final particle-particle interaction, *i.e.* both to the user supplied function and the PME Mesh correction part.

coulomb-modifier

Potential-shift-Verlet

Selects Potential-shift with the Verlet cutoff-scheme, as it is (nearly) free; selects None with the group cutoff-scheme.

Potential-shift

Shift the Coulomb potential by a constant such that it is zero at the cut-off. This makes the potential the integral of the force. Note that this does not affect the forces or the sampling.

None

Use an unmodified Coulomb potential. With the group scheme this means no exact cut-off is used, energies and forces are calculated for all pairs in the pair list.

rcoulomb-switch

(0) [nm] where to start switching the Coulomb potential, only relevant when force or potential switching is used

rcoulomb

(1) [nm] distance for the Coulomb cut-off

epsilon-r

(1) The relative dielectric constant. A value of 0 means infinity.

epsilon-rf

(0) The relative dielectric constant of the reaction field. This is only used with reaction-field electrostatics. A value of 0 means infinity.

Van der Waals

vdwtype

Cut-off

Plain cut-off with pair list radius rlist and VdW cut-off rvdw, where rlist >= rvdw.

PME

Fast smooth Particle-mesh Ewald (SPME) for VdW interactions. The grid dimensions are controlled with fourierspacing in the same way as for electrostatics, and the interpolation order is controlled with pme-order. The relative accuracy of direct/reciprocal space is controlled by ewald-rtol-lj, and the specific combination rules that are to be used by the reciprocal routine are set using lj-pme-comb-rule.

Shift

This functionality is deprecated and replaced by vdw-modifier = Force-switch. The LJ (not Buckingham) potential is decreased over the whole range and the forces decay smoothly to zero between rvdw-switch and rvdw. The neighbor search cut-off rlist should be 0.1 to 0.3 nm larger than rvdw to accommodate for the size of charge groups and diffusion between neighbor list updates.

Switch

This functionality is deprecated and replaced by vdw-modifier = Potential-switch. The LJ (not Buckingham) potential is normal out to rvdw-switch, after which it is switched off to reach zero at rvdw. Both the potential and force functions are continuously smooth, but be aware that all switch functions will give rise to a bulge (increase) in the force (since we are switching the potential). The neighbor search cut-off rlist should be 0.1 to 0.3 nm larger than rvdw to accommodate for the size of charge groups and diffusion between neighbor list updates.

Encad-Shift

The LJ (not Buckingham) potential is decreased over the whole range, using the definition from the Encad simulation package.

User

See user for coulombtype. The function value at zero is not important. When you want to use LJ correction, make sure that rvdw corresponds to the cut-off in the user-defined function. When coulombtype is not set to User the values for the f and -f' columns are ignored.

vdw-modifier

Potential-shift-Verlet

Selects Potential-shift with the Verlet cutoff-scheme, as it is (nearly) free; selects None with the group cutoff-scheme.

Potential-shift

Shift the Van der Waals potential by a constant such that it is zero at the cut-off. This makes the potential the integral of the force. Note that this does not affect the forces or the sampling.

None

Use an unmodified Van der Waals potential. With the group scheme this means no exact cut-off is used, energies and forces are calculated for all pairs in the pair list.

Force-switch

Smoothly switches the forces to zero between rvdw-switch and rvdw. This shifts the potential shift over the whole range and switches it to zero at the cut-off. Note that this is more expensive to calculate than a plain cut-off and it is not required for energy conservation, since Potential-shift conserves energy just as well.

Potential-switch

Smoothly switches the potential to zero between rvdw-switch and rvdw. Note that this introduces artificially large forces in the switching region and is much more expensive to calculate. This option should only be used if the force field you are using requires this.

rvdw-switch

0. [nm]

where to start switching the LJ force and possibly the potential, only relevant when force or potential switching is used

rvdw

(1) [nm] distance for the LJ or Buckingham cut-off

DispCorr

no

don't apply any correction

EnerPres

apply long range dispersion corrections for Energy and Pressure

Ener

apply long range dispersion corrections for Energy only

Tables

table-extension

(1) [nm] Extension of the non-bonded potential lookup tables beyond the largest cut-off distance. The value should be large enough to account for charge group sizes and the diffusion between neighbor-list updates. Without user defined potential the same table length is used for the lookup tables for the 1-4 interactions, which are always tabulated irrespective of the use of tables for the non-bonded interactions. The value of table-extension in no way affects the values of rlist, rcoulomb, or rvdw.

energygrp-table

When user tables are used for electrostatics and/or VdW, here one can give pairs of energy groups for which seperate user tables should be used. The two energy groups will be appended to the table file name, in order of their definition in energygrps, seperated by underscores. For example, if energygrps = Na Cl Sol and energygrp-table = Na Na Na Cl, gmx mdrun will read table_Na_Na.xvg and table_Na_Cl.xvg in addition to the normal table.xvg which will be used for all other energy group pairs.

Pull along an angle defined by four groups. The angle is defined as the angle between two vectors: the vector connecting the COM of the first group to the COM of the second group and the vector connecting the COM of the third group to the COM of the fourth group.

`angle-axis`

As `pull-coord1-geometry=angle` but the second vector is given by `pull-coord1-vec`. Thus, only the two groups that define the first vector need to be given.

`dihedral`

Pull along a dihedral angle defined by six groups. These pairwise define three vectors: the vector connecting the COM of group 1 to the COM of group 2, the COM of group 3 to the COM of group 4, and the COM of group 5 to the COM group 6. The dihedral angle is then defined as the angle between two planes: the plane spanned by the the two first vectors and the plane spanned the two last vectors.

`pull-coord1-groups`

The group indices on which this pull coordinate will operate. The number of group indices required is geometry dependent. The first index can be 0, in which case an absolute reference of `pull-coord1-origin` is used. With an absolute reference the system is no longer translation invariant and one should think about what to do with the center of mass motion.

`pull-coord1-dim`

(Y Y Y) Selects the dimensions that this pull coordinate acts on and that are printed to the output files when `pull-print-components` = `pull-coord1-start=yes`. With `pull-coord1-geometry` = `pull-coord1-geometry=distance`, only Cartesian components set to Y contribute to the distance. Thus setting this to Y Y N results in a distance in the x/y plane. With other geometries all dimensions with non-zero entries in `pull-coord1-vec` should be set to Y, the values for other dimensions only affect the output.

`pull-coord1-origin`

(0.0 0.0 0.0) The pull reference position for use with an absolute reference.

`pull-coord1-vec`

(0.0 0.0 0.0) The pull direction. gmx grompp normalizes the vector.

`pull-coord1-start`

`no`

do not modify `pull-coord1-init`

`yes`

add the COM distance of the starting conformation to `pull-coord1-init`

`pull-coord1-init`

(0.0) [nm] / [deg] The reference distance at t=0.

`pull-coord1-rate`

(0) [nm/ps] / [deg/ps] The rate of change of the reference position.

`pull-coord1-k`

(0) [kJ mol-1 nm-2] / [kJ mol-1 nm-1] / [kJ mol-1 rad-2] / [kJ mol-1 rad-1] The force constant. For umbrella pulling this is the harmonic force constant in kJ mol-1 nm-2 (or kJ mol-1 rad-2 for angles). For constant force pulling this is the force constant of the linear potential, and thus the negative (!) of the constant force in kJ mol-1 nm-1 (or kJ mol-1 rad-1 for angles). Note that for angles the force constant is expressed in terms of radians (while `pull-coord1-init` and `pull-coord1-rate` are expressed in degrees).

`pull-coord1-kB`

(pull-k1) [kJ mol-1 nm-2] / [kJ mol-1 nm-1] / [kJ mol-1 rad-2] / [kJ mol-1 rad-1] As `pull-coord1-k`, but for state B. This is only used when `free-energy` is turned on. The force constant is then (1 - lambda) * `pull-coord1-k` + lambda * `pull-coord1-kB`.

AWH adaptive biasing

`awh`

`no`

No biasing.

`yes`

Adaptively bias a reaction coordinate using the AWH method and estimate the corresponding PMF. The PMF and other AWH data are written to energy file at an interval set by `awh-nstout` and can be extracted with the `gmx awh` tool. The AWH coordinate can be multidimensional and is defined by mapping each dimension to a pull coordinate index. This is only allowed if `pull-coord1-type=external-potential` and `pull-coord1-potential-provider` = `awh` for the concerned pull coordinate indices.

`awh-potential`

`convolved`

The applied biasing potential is the convolution of the bias function and a set of harmonic umbrella potentials (see `awh-potential=umbrella` below). This results in a smooth potential function and force. The resolution of the potential is set by the force constant of each umbrella, see `awh1-dim1-force-constant`.

`umbrella`

The potential bias is applied by controlling the position of an harmonic potential using Monte-Carlo sampling. The force constant is set with `awh1-dim1-force-constant`. The umbrella location is sampled using Monte-Carlo every `awh-nstsample` steps. There are no advantages to using an umbrella. This option is mainly for comparison and testing purposes.

`awh-share-multisim`

`no`

AWH will not share biases across simulations started with gmx mdrun option `-multidir`. The biases will be independent.

`yes`

With gmx mdrun and option `-multidir` the bias and PMF estimates for biases with `awh1-share-group` >0 will be shared across simulations with the biases with the same `awh1-share-group` value. The simulations should have the same AWH settings for sharing to make sense. gmx mdrun will check whether the simulations are technically compatible for sharing, but the user should check that bias sharing physically makes sense.

`awh-seed`

(-1) Random seed for Monte-Carlo sampling the umbrella position, where -1 indicates to generate a seed. Only used with `awh-potential=umbrella`.

`awh-nstout`

(100000) Number of steps between printing AWH data to the energy file, should be a multiple of `nstenergy`.

`awh-nstsample`

(10) Number of steps between sampling of the coordinate value. This sampling is the basis for updating the bias and estimating the PMF and other AWH observables.

`awh-nsamples-update`

(10) The number of coordinate samples used for each AWH update. The update interval in steps is `awh-nstsample` times this value.

`awh-nbias`

(1) The number of biases, each acting on its own coordinate. The following options should be specified for each bias although below only the options for bias number 1 is shown. Options for other bias indices are obtained by replacing '1' by the bias index.

`awh1-error-init`

(10.0) [kJ mol-1] Estimated initial average error of the PMF for this bias. This value together with the given diffusion constant(s) `awh1-dim1-diffusion` determine the initial biasing rate. The error is obviously not known *a priori*. Only a rough estimate of `awh1-error-init` is needed however. As a general guideline, leave `awh1-error-init` to its default value when starting a new simulation. On the other hand, when there is *a priori* knowledge of the PMF (e.g. when an initial PMF estimate is provided, see the `awh1-user-data` option) then `awh1-error-init` should reflect that knowledge.

`awh1-growth`

`exp-linear`

Each bias keeps a reference weight histogram for the coordinate samples. Its size sets the magnitude of the bias function and free energy estimate updates (few samples corresponds to large updates and vice versa). Thus, its growth rate sets the maximum convergence rate. By default, there is an initial stage in which the histogram grows close to exponentially (but slower than the sampling rate). In the final stage that follows, the growth rate is linear and equal to the sampling rate (set by `awh-nstsample`). The initial stage is typically necessary for efficient convergence when starting a new simulation where high free energy barriers have not yet been flattened by the bias.

`linear`

As `awh1-growth=exp-linear` but skip the initial stage. This may be useful if there is *a priori* knowledge (see `awh1-error-init`) which eliminates the need for an initial stage. This is also the setting compatible with `awh1-target=local-boltzmann`.

`awh1-equilibrate-histogram`

`no`

Do not equilibrate histogram.

`yes`

Before entering the initial stage (see `awh1-growth=exp-linear`), make sure the histogram of sampled weights is following the target distribution closely enough (specifically, at least 80% of the target region needs to have a local relative error of less than 20%). This option would typically only be used when `awh1-share-group` > 0 and the initial configurations poorly represent the target distribution.

`awh1-target`

`constant`

The bias is tuned towards a constant (uniform) coordinate distribution in the defined sampling interval (defined by [`awh1-dim1-start`, `awh1-dim1-end`]).

`cutoff`

Similar to `awh1-target=constant`, but the target distribution is proportional to 1/(1 + exp(F - `awh1-target=cutoff`)), where F is the free energy relative to the estimated global minimum. This provides a smooth switch of a flat target distribution in regions with free energy lower than the cut-off to a Boltzmann distribution in regions with free energy higher than the cut-off.

`boltzmann`

The target distribution is a Boltzmann distribtution with a scaled beta (inverse temperature) factor given by `awh1-target-beta-scaling`. *E.g.*, a value of 0.1 would give the same coordinate distribution as sampling with a simulation temperature scaled by 10.

`local-boltzmann`

Same target distribution and use of `awh1-target-beta-scaling` but the convergence towards the target distribution is inherently local *i.e.*, the rate of change of the bias only depends on the local sampling. This local convergence property is only compatible with `awh1-growth=linear`, since for `awh1-growth=exp-linear` histograms are globally rescaled in the initial stage.

`awh1-target-beta-scaling`

[0] [] For `awh1-target=boltzmann` and `awh1-target=local-boltzmann` it is the unitless beta scaling factor taking values in (0,1).

`awh1-target-cutoff`

[0] [kJ mol-1] For `awh1-target=cutoff` this is the cutoff, should be > 0.

`awh1-user-data`

`no`

Initialize the PMF and target distribution with default values.

`yes`

Initialize the PMF and target distribution with user provided data. For `awh-nbias` = 1, gmx mdrun will expect a file `awhinit.xvg` to be present in the run directory. For multiple biases, gmx mdrun expects files `awhinit1.xvg`, `awhinit2.xvg`, etc. The file name can be changed with the `-awh` option. The first `awh1-ndim` columns of each input file should contain the coordinate values, such that each row defines a point in coordinate space. Column `awh1-ndim` + 1 should contain the PMF value for each point. The target distribution column can either follow the PMF (column `awh1-ndim` + 2) or be in the same column as written by gmx awh.

`awh1-share-group`

`0`

Do not share the bias.

`positive`

Share the bias and PMF estimates within and/or between simulations. Within a simulation, the bias will be shared between biases that have the same `awh1-share-group` index (note that the current code does not support this). With `awh-share-multisim=yes` and gmx mdrun option `-multidir` the bias will also be shared across simulations. Sharing may increase convergence initially, although the starting configurations can be critical, especially when sharing between many biases. Currently, positive group values should start at 1 and increase by 1 for each subsequent bias that is shared.

`awh1-ndim`

(1) [integer] Number of dimensions of the coordinate, each dimension maps to 1 pull coordinate. The following options should be specified for each such dimension. Below only the options for dimension number 1 is shown. Options for other dimension indices are obtained by replacing '1' by the dimension index.

`awh1-dim1-coord-provider`

`pull`

The module providing the reaction coordinate for this dimension. Currently AWH can only act on pull coordinates.

`awh1-dim1-coord-index`

(1) Index of the pull coordinate defining this coordinate dimension.

`awh1-dim1-force-constant`

(0) [kJ/mol/nm^2] or [kJ/mol/rad^2] Force constant for the (convolved) umbrella potential(s) along this coordinate dimension.

`awh1-dim1-start`

(0.0) [nm]/[rad] Start value of the sampling interval along this dimension. The range of allowed values depends on the relevant pull geometry (see `pull-coord1-geometry`). For periodic geometries `awh1-dim1-start` greater than `awh1-dim1-end` is allowed. The interval will then wrap around from +period/2 to -period/2.

`awh1-dim1-end`

(0.0) [nm]/[rad] End value defining the sampling interval together with `awh1-dim1-start`.

`awh1-dim1-period`

(0.0) [nm]/[rad] The period of this reaction coordinate, use 0 when the coordinate is not periodic.

`awh1-dim1-diffusion`

(1e-5) [nm^2/ps]/[rad^2/ps] Estimated diffusion constant for this coordinate dimension determining the initial biasing rate. This needs only be a rough estimate and should not critically affect the results unless it is set to something very low, leading to slow convergence, or very high, forcing the system far from equilibrium. Not setting this value explicitly generates a warning.

`awh1-dim1-cover-diameter`

(0.0)) [nm]/[rad] Diameter that needs to be sampled by a single simulation around a coordinate value before the point is considered covered in the initial stage (see `awh1-growth=exp-linear`). A value > 0 ensures that for each covering there is a continuous transition of this diameter across each coordinate value. This is trivially true for independent simulations but not for for multiple bias-sharing simulations (`awh1-share-group`>0). For a diameter = 0, covering occurs as soon as the simulations have sampled the whole interval, which for many sharing simulations does not guarantee transitions across free energy barriers. On the other hand, when the diameter >= the sampling interval length, covering occurs when a single simulation has independently sampled the whole interval.

Enforced rotation

These mdp parameters can be used enforce the rotation of a group of atoms, e.g. a protein subunit. The reference manual describes in detail 13 different potentials that can be used to achieve such a rotation.

`rotation`

`no`

No enforced rotation will be applied. All enforced rotation options will be ignored (and if present in the mdp file, they unfortunately generate warnings).

`yes`

Apply the rotation potential specified by `rot-type0` to the group of atoms given under the `rot-group0` option.

`rot-ngroups`

(1) Number of rotation groups.

`rot-group0`

Name of rotation group 0 in the index file.

`rot-type0`

(iso) Type of rotation potential that is applied to rotation group 0. Can be of of the following: `iso`, `iso-pf`, `pm`, `pm-pf`, `rm`, `rm-pf`, `rm2`, `rm2-pf`, `flex`, `flex-t`, `flex2`, or `flex2-t`.

`rot-massw0`

(no) Use mass weighted rotation group positions.

`rot-vec0`

(1.0 0.0 0.0) Rotation vector, will get normalized.

`rot-pivot0`

(0.0 0.0 0.0) Pivot point (nm) for the potentials `iso`, `pm`, `rm`, and `rm2`.

`rot-rate0`

(0) Reference rotation rate (degree/ps) of group 0.

`rot-k0`

(0) Force constant (kJ/(mol*nm^2)) for group 0.

`rot-slab-dist0`

(1.5) Slab distance (nm), if a flexible axis rotation type was chosen.

`rot-min-gauss0`

(0.001) Minimum value (cutoff) of Gaussian function for the force to be evaluated (for the flexible axis potentials).

`rot-eps0`

(0.0001) Value of additive constant epsilon' (nm^2) for `rm2*` and `flex2*` potentials.

`rot-fit-method0`

(rmsd) Fitting method when determining the actual angle of a rotation group (can be one of `rmsd`, `norm`, or `potential`).

`rot-potfit-nsteps0`

(21) For fit type `potential`, the number of angular positions around the reference angle for which the rotation potential is evaluated.

`rot-potfit-step0`

(0.25) For fit type `potential`, the distance in degrees between two angular positions.

`rot-nstrout`

(100) Output frequency (in steps) for the angle of the rotation group, as well as for the torque and the rotation potential energy.

`rot-nstsout`

(1000) Output frequency for per-slab data of the flexible axis potentials, i.e. angles, torques and slab centers.

NMR refinement

`disre`

`no`

ignore distance restraint information in topology file

`simple`

simple (per-molecule) distance restraints.

`ensemble`

distance restraints over an ensemble of molecules in one simulation box. Normally, one would perform ensemble averaging over multiple subsystems, each in a separate box, using `mdrun -multi`. Supply `topol0.tpr`, `topol1.tpr`, … with different coordinates and/or velocities. The environment variable `GMX_DISRE_ENSEMBLE_SIZE` sets the number of systems within each ensemble (usually equal to the `mdrun -multi` value).

`disre-weighting`

`equal`

divide the restraint force equally over all atom pairs in the restraint

`conservative`

the forces are the derivative of the restraint potential, this results in an weighting of the atom pairs to the reciprocal seventh power of the displacement. The forces are conservative when `disre-tau` is zero.

`disre-mixed`

`no`

the violation used in the calculation of the restraint force is the time-averaged violation

`yes`

the violation used in the calculation of the restraint force is the square root of the product of the time-averaged violation and the instantaneous violation

`disre-fc`

(1000) [kJ mol-1 nm-2] force constant for distance restraints, which is multiplied by a (possibly) different factor for each restraint given in the *fac* column of the interaction in the topology file.

`disre-tau`

(0) [ps] time constant for distance restraints running average. A value of zero turns off time averaging.

`nstdisreout`

(100) [steps] period between steps when the running time-averaged and instantaneous distances of all atom pairs involved in restraints are written to the energy file (can make the energy file very large)

`orire`

`no`

ignore orientation restraint information in topology file

`yes`

use orientation restraints, ensemble averaging can be performed with *mdrun -multi*

`orire-fc`

(0) [kJ mol] force constant for orientation restraints, which is multiplied by a (possibly) different weight factor for each restraint, can be set to zero to obtain the orientations from a free simulation

`orire-tau`

(0) [ps] time constant for orientation restraints running average. A value of zero turns off time averaging.

`orire-fitgrp`

fit group for orientation restraining. This group of atoms is used to determine the rotation **R** of the system with respect to the reference orientation. The reference orientation is the starting conformation of the first subsystem. For a protein, backbone is a reasonable choice

`nstorireout`

(100) [steps] period between steps when the running time-averaged and instantaneous orientations for all restraints, and the molecular order tensor are written to the energy file (can make the energy file very large)

Free energy calculations

`free-energy`

`no`

Only use topology A.

`yes`

Interpolate between topology A (lambda=0) to topology B (lambda=1) and write the derivative of the Hamiltonian with respect to lambda (as specified with `dhdl-derivatives`), or the Hamiltonian differences with respect to other lambda values (as specified with foreign lambda) to the energy file and/or to `dhdl.xvg`, where they can be processed by, for example gmx bar. The potentials, bond-lengths and angles are interpolated linearly as described in the manual. When `sc-alpha` is larger than zero, soft-core potentials are used for the LJ and Coulomb interactions.

`expanded`

Turns on expanded ensemble simulation, where the alchemical state becomes a dynamic variable, allowing jumping between different Hamiltonians. See the expanded ensemble options for controlling how expanded ensemble simulations are performed. The different Hamiltonians used in expanded ensemble simulations are defined by the other free energy options.

`init-lambda`

(-1) starting value for lambda (float). Generally, this should only be used with slow growth (*i.e.* nonzero `delta-lambda`). In other cases, `init-lambda-state` should be specified instead. Must be greater than or equal to 0.

`delta-lambda`

(0) increment per time step for lambda

`init-lambda-state`

(-1) starting value for the lambda state (integer). Specifies which columm of the lambda vector (`coul-lambdas`, `vdw-lambdas`, `bonded-lambdas`, `restraint-lambdas`, `mass-lambdas`, `temperature-lambdas`, `fep-lambdas`) should be used. This is a zero-based index: `init-lambda-state` 0 means the first column, and so on.

`fep-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Free energy differences between different lambda values can then be determined with gmx bar. `fep-lambdas` is different from the other -lambdas keywords because all components of the lambda vector that are not specified will use `fep-lambdas` (including `restraint-lambdas` and therefore the pull code restraints).

`coul-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the electrostatic interactions are controlled with this component of the lambda vector (and only if the lambda=0 and lambda=1 states have differing electrostatic interactions).

`vdw-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the van der Waals interactions are controlled with this component of the lambda vector.

`bonded-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the bonded interactions are controlled with this component of the lambda vector.

`restraint-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the restraint interactions: dihedral restraints, and the pull code restraints are controlled with this component of the lambda vector.

`mass-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the particle masses are controlled with this component of the lambda vector.

`temperature-lambdas`

[array] Zero, one or more lambda values for which Delta H values will be determined and written to dhdl.xvg every `nstdhdl` steps. Values must be between 0 and 1. Only the temperatures controlled with this component of the lambda vector. Note that these lambdas should not be used for replica exchange, only for simulated tempering.

`calc-lambda-neighbors`

(1) Controls the number of lambda values for which Delta H values will be calculated and written out, if `init-lambda-state` has been set. A positive value will limit the number of lambda points calculated to only the nth neighbors of `init-lambda-state`: for example, if `init-lambda-state` is 5 and this parameter has a value of 2, energies for lambda points 3-7 will be calculated and writen out. A value of -1 means all lambda points will be written out. For normal BAR such as with gmx bar, a value of 1 is sufficient, while for MBAR -1 should be used.

`sc-alpha`

(0) the soft-core alpha parameter, a value of 0 results in linear interpolation of the LJ and Coulomb interactions

`sc-r-power`

(6) the power of the radial term in the soft-core equation. Possible values are 6 and 48. 6 is more standard, and is the default. When 48 is used, then sc-alpha should generally be much lower (between 0.001 and 0.003).

`sc-coul`

(no) Whether to apply the soft-core free energy interaction transformation to the Columbic interaction of a molecule. Default is no, as it is generally more efficient to turn off the Coulomic interactions linearly before turning off the van der Waals interactions. Note that it is only taken into account when lambda states are used, not with `couple-lambda0` / `couple-lambda1`, and you can still turn off soft-core interactions by setting `sc-alpha` to 0.

`sc-power`

(0) the power for lambda in the soft-core function, only the values 1 and 2 are supported

`sc-sigma`

(0.3) [nm] the soft-core sigma for particles which have a C6 or C12 parameter equal to zero or a sigma smaller than `sc-sigma`

`couple-moltype`

Here one can supply a molecule type (as defined in the topology) for calculating solvation or coupling free energies. There is a special option `system` that couples all molecule types in the system. This can be useful for equilibrating a system starting from (nearly) random coordinates. `free-energy` has to be turned on. The Van der Waals interactions and/or charges in this molecule type can be turned on or off between lambda=0 and lambda=1, depending on the settings of `couple-lambda0` and `couple-lambda1`. If you want to decouple one of several copies of a molecule, you need to copy and rename the molecule definition in the topology.

`couple-lambda0`

`vdw-q`

all interactions are on at lambda=0

`vdw`

the charges are zero (no Coulomb interactions) at lambda=0

`q`

the Van der Waals interactions are turned at lambda=0; soft-core interactions will be required to avoid singularities

`none`

the Van der Waals interactions are turned off and the charges are zero at lambda=0; soft-core interactions will be required to avoid singularities.

`couple-lambda1`

analogous to `couple-lambda1`, but for lambda=1

`couple-intramol`

`no`

All intra-molecular non-bonded interactions for moleculetype `couple-moltype` are replaced by exclusions and explicit pair interactions. In this manner the decoupled state of the molecule corresponds to the proper vacuum state without periodicity effects.

`yes`

The intra-molecular Van der Waals and Coulomb interactions are also turned on/off. This can be useful for partitioning free-energies of relatively large molecules, where the intra-molecular non-bonded interactions might lead to kinetically trapped vacuum conformations. The 1-4 pair interactions are not turned off.

`nstdhdl`

(100) the frequency for writing dH/dlambda and possibly Delta H to dhdl.xvg, 0 means no ouput, should be a multiple of `nstcalcenergy`.

`dhdl-derivatives`

(yes)

If yes (the default), the derivatives of the Hamiltonian with respect to lambda at each `nstdhdl` step are written

out. These values are needed for interpolation of linear energy differences with gmx bar (although the same can also be achieved with the right foreign lambda setting, that may not be as flexible), or with thermodynamic integration

`dhdl-print-energy`

(no)

Include either the total or the potential energy in the dhdl file. Options are ‘no’, ‘potential’, or ‘total’. This information is needed for later free energy analysis if the states of interest are at different temperatures. If all states are at the same temperature, this information is not needed. ‘potential’ is useful in case one is using `mdrun -rerun` to generate the `dhdl.xvg` file. When rerunning from an existing trajectory, the kinetic energy will often not be correct, and thus one must compute the residual free energy from the potential alone, with the kinetic energy component computed analytically.

`separate-dhdl-file`

`yes`

The free energy values that are calculated (as specified with the foreign lambda and `dhdl-derivatives` settings) are written out to a separate file, with the default name `dhdl.xvg`. This file can be used directly with gmx bar.

`no`

The free energy values are written out to the energy output file (`ener.edr`, in accumulated blocks at every `nstenergy` steps), where they can be extracted with gmx energy or used directly with gmx bar.

`dh-hist-size`

(0) If nonzero, specifies the size of the histogram into which the Delta H values (specified with foreign lambda) and the derivative dH/dl values are binned, and written to ener.edr. This can be used to save disk space while calculating free energy differences. One histogram gets written for each foreign lambda and two for the dH/dl, at every `nstenergy` step. Be aware that incorrect histogram settings (too small size or too wide bins) can introduce errors. Do not use histograms unless you’re certain you need it.

`dh-hist-spacing`

(0.1) Specifies the bin width of the histograms, in energy units. Used in conjunction with `dh-hist-size`. This size limits the accuracy with which free energies can be calculated. Do not use histograms unless you’re certain you need it.

Expanded Ensemble calculations

`nstexpanded`

The number of integration steps beween attempted moves changing the system Hamiltonian in expanded ensemble simulations. Must be a multiple of `nstcalcenergy`, but can be greater or less than `nstdhdl`.

`lmc-stats`

`no`

No Monte Carlo in state space is performed.

`metropolis-transition`

Uses the Metropolis weights to update the expanded ensemble weight of each state. Min{1,exp(-(beta_new u_new - beta_old u_old)}

`barker-transition`

Uses the Barker transition critera to update the expanded ensemble weight of each state i, defined by exp(-beta_new u_new)/(exp(-beta_new u_new)+exp(-beta_old u_old))

`wang-landau`

Uses the Wang-Landau algorithm (in state space, not energy space) to update the expanded ensemble weights.

`min-variance`

Uses the minimum variance updating method of Escobedo et al. to update the expanded ensemble weights. Weights will not be the free energies, but will rather emphasize states that need more sampling to give even uncertainty.

`lmc-mc-move`

`no`

No Monte Carlo in state space is performed.

`metropolis-transition`

Randomly chooses a new state up or down, then uses the Metropolis critera to decide whether to accept or reject: Min{1,exp(-(beta_new u_new - beta_old u_old)}

`barker-transition`

Randomly chooses a new state up or down, then uses the Barker transition critera to decide whether to accept or reject: exp(-beta_new u_new)/(exp(-beta_new u_new)+exp(-beta_old u_old))

`gibbs`

Uses the conditional weights of the state given the coordinate (exp(-beta_i u_i) / sum_k exp(beta_i u_i) to decide which state to move to.

`metropolized-gibbs`

Uses the conditional weights of the state given the coordinate (exp(-beta_i u_i) / sum_k exp(beta_i u_i) to decide which state to move to, EXCLUDING the current state, then uses a rejection step to ensure detailed balance. Always more efficient that Gibbs, though only marginally so in many situations, such as when only the nearest neighbors have decent phase space overlap.

`lmc-seed`

(-1) random seed to use for Monte Carlo moves in state space. When `lmc-seed` is set to -1, a pseudo random seed is us

`mc-temperature`

Temperature used for acceptance/rejection for Monte Carlo moves. If not specified, the temperature of the simulation specified in the first group of `ref-t` is used.

`wl-ratio`

(0.8) The cutoff for the histogram of state occupancies to be reset, and the free energy incrementor to be changed from delta to delta * `wl-scale`. If we define the Nratio = (number of samples at each histogram) / (average number of samples at each histogram). `wl-ratio` of 0.8 means that means that the histogram is only considered flat if all Nratio > 0.8 AND simultaneously all 1/Nratio > 0.8.

`wl-scale`

(0.8) Each time the histogram is considered flat, then the current value of the Wang-Landau incrementor for the free energies is multiplied by `wl-scale`. Value must be between 0 and 1.

`init-wl-delta`

(1.0) The initial value of the Wang-Landau incrementor in kT. Some value near 1 kT is usually most efficient, though sometimes a value of 2-3 in units of kT works better if the free energy differences are large.

`wl-oneovert`

(no) Set Wang-Landau incrementor to scale with 1/(simulation time) in the large sample limit. There is significant evidence that the standard Wang-Landau algorithms in state space presented here result in free energies getting ‘burned in’ to incorrect values that depend on the initial state. when `wl-oneovert` is true, then when the incrementor becomes less than 1/N, where N is the mumber of samples collected (and thus proportional to the data collection time, hence ‘1 over t’), then the Wang-Lambda incrementor is set to 1/N, decreasing every step. Once this occurs, `wl-ratio` is ignored, but the weights will still stop updating when the equilibration criteria set in `lmc-weights-equil` is achieved.

`lmc-repeats`

(1) Controls the number of times that each Monte Carlo swap type is performed each iteration. In the limit of large numbers of Monte Carlo repeats, then all methods converge to Gibbs sampling. The value will generally not need to be different from 1.

`lmc-gibbsdelta`

(-1) Limit Gibbs sampling to selected numbers of neighboring states. For Gibbs sampling, it is sometimes inefficient to perform Gibbs sampling over all of the states that are defined. A positive value of `lmc-gibbsdelta` means that only states plus or minus `lmc-gibbsdelta` are considered in exchanges up and down. A value of -1 means that all states are considered. For less than 100 states, it is probably not that expensive to include all states.

`lmc-forced-nstart`

(0) Force initial state space sampling to generate weights. In order to come up with reasonable initial weights, this setting allows the simulation to drive from the initial to the final lambda state, with `lmc-forced-nstart` steps at each state before moving on to the next lambda state. If `lmc-forced-nstart` is sufficiently long (thousands of steps, perhaps), then the weights will be close to correct. However, in most cases, it is probably better to simply run the standard weight equilibration algorithms.

`nst-transition-matrix`

(-1) Frequency of outputting the expanded ensemble transition matrix. A negative number means it will only be printed at the end of the simulation.

`symmetrized-transition-matrix`

(no) Whether to symmetrize the empirical transition matrix. In the infinite limit the matrix will be symmetric, but will diverge with statistical noise for short timescales. Forced symmetrization, by using the matrix T_sym = 1/2 (T + transpose(T)), removes problems like the existence of (small magnitude) negative eigenvalues.

`mininum-var-min`

(100) The min-variance strategy (option of `lmc-stats` is only valid for larger number of samples, and can get stuck if too few samples are used at each state. `mininum-var-min` is the minimum number of samples that each state that are allowed before the min-variance strategy is activated if selected.

`init-lambda-weights`

The initial weights (free energies) used for the expanded ensemble states. Default is a vector of zero weights. format is similar to the lambda vector settings in `fep-lambdas`, except the weights can be any floating point number. Units are kT. Its length must match the lambda vector lengths.

`lmc-weights-equil`

`no`

Expanded ensemble weights continue to be updated throughout the simulation.

`yes`

The input expanded ensemble weights are treated as equilibrated, and are not updated throughout the simulation.

`wl-delta`

Expanded ensemble weight updating is stopped when the Wang-Landau incrementor falls below this value.

`number-all-lambda`

Expanded ensemble weight updating is stopped when the number of samples at all of the lambda states is greater than this value.

`number-steps`

Expanded ensemble weight updating is stopped when the number of steps is greater than the level specified by this value.

`number-samples`

Expanded ensemble weight updating is stopped when the number of total samples across all lambda states is greater than the level specified by this value.

`count-ratio`

Expanded ensemble weight updating is stopped when the ratio of samples at the least sampled lambda state and most sampled lambda state greater than this value.

`simulated-tempering`

(no) Turn simulated tempering on or off. Simulated tempering is implemented as expanded ensemble sampling with different temperatures instead of different Hamiltonians.

`sim-temp-low`

(300) [K] Low temperature for simulated tempering.

`sim-temp-high`

(300) [K] High temperature for simulated tempering.

`simulated-tempering-scaling`

Controls the way that the temperatures at intermediate lambdas are calculated from the `temperature-lambdas` part of the lambda vector.

`linear`

Linearly interpolates the temperatures using the values of `temperature-lambdas`, *i.e.* if `sim-temp-low` =300, `sim-temp-high` =400, then lambda=0.5 correspond to a temperature of 350. A nonlinear set of temperatures can always be implemented with uneven spacing in lambda.

`geometric`

Interpolates temperatures geometrically between `sim-temp-low` and `sim-temp-high`. The i:th state has temperature `sim-temp-low` * (`sim-temp-high` / `sim-temp-low`) raised to the power of (i/(ntemps-1)). This should give roughly equal exchange for constant heat capacity, though of course things simulations that involve protein folding have very high heat capacity peaks.

`exponential`

Interpolates temperatures exponentially between `sim-temp-low` and `sim-temp-high`. The i:th state has temperature `sim-temp-low` + (`sim-temp-high` - `sim-temp-low`)*((exp(`temperature-lambdas` (i))-1)/(exp(1.0)-i)).

Non-equilibrium MD

`acc-grps`

groups for constant acceleration (*e.g.* `Protein Sol`) all atoms in groups Protein and Sol will experience constant acceleration as specified in the `accelerate` line

`accelerate`

(0) [nm ps^-2] acceleration for `acc-grps`; x, y and z for each group (*e.g.* `0.1 0.0 0.0 -0.1 0.0 0.0` means that first group has constant acceleration of 0.1 nm ps-2 in X direction, second group the opposite).

`freezegrps`

Groups that are to be frozen (*i.e.* their X, Y, and/or Z position will not be updated; *e.g.* `Lipid SOL`). `freezedim` specifies for which dimension the freezing applies. To avoid spurious contibrutions to the virial and pressure due to large forces between completely frozen atoms you need to use energy group exclusions, this also saves computing time. Note that coordinates of frozen atoms are not scaled by pressure-coupling algorithms.

`freezedim`

dimensions for which groups in `freezegrps` should be frozen, specify *Y* or *N* for X, Y and Z and for each group (*e.g.* `Y Y N N N N` means that particles in the first group can move only in Z direction. The particles in the second group can move in any direction).

`cos-acceleration`

(0) [nm ps^-2] the amplitude of the acceleration profile for calculating the viscosity. The acceleration is in the X-direction and the magnitude is `cos-acceleration` cos(2 pi z/boxheight). Two terms are added to the energy file: the amplitude of the velocity profile and 1/viscosity.

`deform`

(0 0 0 0 0 0) [nm ps-1] The velocities of deformation for the box elements: a(x) b(y) c(z) b(x) c(x) c(y). Each step the box elements for which `deform` is non-zero are calculated as: box(ts)+(t-ts)*deform, off-diagonal elements are corrected for periodicity. The coordinates are transformed accordingly. Frozen degrees of freedom are (purposely) also transformed. The time ts is set to t at the first step and at steps at which x and v are written to trajectory to ensure exact restarts. Deformation can be used together with semiisotropic or anisotropic pressure coupling when the appropriate compressibilities are set to zero. The diagonal elements can be used to strain a solid. The off-diagonal elements can be used to shear a solid or a liquid.

Electric fields

`electric-field-x ; electric-field-y ; electric-field-z`

Here you can specify an electric field that optionally can be alternating and pulsed. The general expression for the field has the form of a gaussian laser pulse:

E(t) = E0 exp (-(t-t0)^2/(2 sigma^2)) cos(omega (t-t0))

For example, the four parameters for direction x are set in the three fields of `electric-field-x` (and similar for y and z) like

electric-field-x = E0 omega t0 sigma

In the special case that sigma = 0, the exponential term is omitted and only the cosine term is used. If also omega = 0 a static electric field is applied.

More details in Carl Caleman and David van der Spoel: Picosecond Melting of Ice by an Infrared Laser Pulse - A Simulation Study Angew. Chem. Intl. Ed. 47 pp. 14 17-1420 (2008)

Mixed quantum/classical molecular dynamics

`QMMM`

`no`

No QM/MM.

`yes`

Do a QM/MM simulation. Several groups can be described at different QM levels separately. These are specified in the `QMMM-grps` field separated by spaces. The level of *ab initio* theory at which the groups are described is specified by `QMmethod` and `QMbasis` Fields. Describing the groups at different levels of theory is only possible with the ONIOM QM/MM scheme, specified by `QMMMscheme`.

`QMMM-grps`

groups to be descibed at the QM level

`QMMMscheme`

`normal`

normal QM/MM. There can only be one `QMMM-grps` that is modelled at the `QMmethod` and `QMbasis` level of *ab initio* theory. The rest of the system is described at the MM level. The QM and MM subsystems interact as follows: MM point charges are included in the QM one-electron hamiltonian and all Lennard-Jones interactions are described at the MM level.

`ONIOM`

The interaction between the subsystem is described using the ONIOM method by Morokuma and co-workers. There can be more than one `QMMM-grps` each modeled at a different level of QM theory (`QMmethod` and `QMbasis`).

`QMmethod`

(RHF) Method used to compute the energy and gradients on the QM atoms. Available methods are AM1, PM3, RHF, UHF, DFT, B3LYP, MP2, CASSCF, and MMVB. For CASSCF, the number of electrons and orbitals included in the active space is specified by `CASelectrons` and `CASorbitals`.

`QMbasis`

(STO-3G) Basis set used to expand the electronic wavefuntion. Only Gaussian basis sets are currently available, *i.e.* `STO-3G, 3-21G, 3-21G*, 3-21+G*, 6-21G, 6-31G, 6-31G*, 6-31+G*,` and `6-311G`.

`QMcharge`

(0) [integer] The total charge in *e* of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the total charge of each ONIOM layer needs to be specified separately.

`QMmult`

(1) [integer] The multiplicity of the `QMMM-grps`. In case there are more than one `QMMM-grps`, the multiplicity of each ONIOM layer needs to be specified separately.

`CASorbitals`

(0) [integer] The number of orbitals to be included in the active space when doing a CASSCF computation.

`CASelectrons`

(0) [integer] The number of electrons to be included in the active space when doing a CASSCF computation.

`SH`

`no`

No surface hopping. The system is always in the electronic ground-state.

`yes`

Do a QM/MM MD simulation on the excited state-potential energy surface and enforce a *diabatic* hop to the ground-state when the system hits the conical intersection hyperline in the course the simulation. This option only works in combination with the CASSCF method.

Implicit solvent

`implicit-solvent`

`no`

No implicit solvent

`GBSA`

Do a simulation with implicit solvent using the Generalized Born formalism. Three different methods for calculating the Born radii are available, Still, HCT and OBC. These are specified with the `gb-algorithm` field. The non-polar solvation is specified with the `sa-algorithm` field.

`gb-algorithm`

`Still`

Use the Still method to calculate the Born radii

`HCT`

Use the Hawkins-Cramer-Truhlar method to calculate the Born radii

`OBC`

Use the Onufriev-Bashford-Case method to calculate the Born radii

`nstgbradii`

(1) [steps] Frequency to (re)-calculate the Born radii. For most practial purposes, setting a value larger than 1 violates energy conservation and leads to unstable trajectories.

`rgbradii`

(1.0) [nm] Cut-off for the calculation of the Born radii. Currently must be equal to rlist

`gb-epsilon-solvent`

(80) Dielectric constant for the implicit solvent

`gb-saltconc`

(0) [M] Salt concentration for implicit solvent models, currently not used

`gb-obc-alpha`

`gb-obc-beta`

`gb-obc-gamma`

Scale factors for the OBC model. Default values of 1, 0.78 and 4.85 respectively are for OBC(II). Values for OBC(I) are 0.8, 0 and 2.91 respectively

`gb-dielectric-offset`

(0.009) [nm] Distance for the di-electric offset when calculating the Born radii. This is the offset between the center of each atom the center of the polarization energy for the corresponding atom

`sa-algorithm`

`Ace-approximation`

Use an Ace-type approximation

`None`

No non-polar solvation calculation done. For GBSA only the polar part gets calculated

`sa-surface-tension`

[kJ mol-1 nm-2] Default value for surface tension with SA algorithms. The default value is -1; Note that if this default value is not changed it will be overridden by gmx grompp using values that are specific for the choice of radii algorithm (0.0049 kcal/mol/Angstrom^2 for Still, 0.0054 kcal/mol/Angstrom2 for HCT/OBC) Setting it to 0 will while using an sa-algorithm other than None means no non-polar calculations are done.

Computational Electrophysiology

Use these options to switch on and control ion/water position exchanges in “Computational Electrophysiology” simulation setups. (See the reference manual for details).

`swapcoords`

`no`

Do not enable ion/water position exchanges.

`X ; Y ; Z`

Allow for ion/water position exchanges along the chosen direction. In a typical setup with the membranes parallel to the x-y plane, ion/water pairs need to be exchanged in Z direction to sustain the requested ion concentrations in the compartments.

`swap-frequency`

(1) The swap attempt frequency, i.e. every how many time steps the ion counts per compartment are determined and exchanges made if necessary. Normally it is not necessary to check at every time step. For typical Computational Electrophysiology setups, a value of about 100 is sufficient and yields a negligible performance impact.

`split-group0`

Name of the index group of the membrane-embedded part of channel #0. The center of mass of these atoms defines one of the compartment boundaries and should be chosen such that it is near the center of the membrane.

`split-group1`

Channel #1 defines the position of the other compartment boundary.

`massw-split0`

(no) Defines whether or not mass-weighting is used to calculate the split group center.

`no`

Use the geometrical center.

`yes`

Use the center of mass.

`massw-split1`

(no) As above, but for split-group #1.

`solvent-group`

Name of the index group of solvent molecules.

`coupl-steps`

(10) Average the number of ions per compartment over these many swap attempt steps. This can be used to prevent that ions near a compartment boundary (diffusing through a channel, e.g.) lead to unwanted back and forth swaps.

`iontypes`

(1) The number of different ion types to be controlled. These are during the simulation exchanged with solvent molecules to reach the desired reference numbers.

`iontype0-name`

Name of the first ion type.

`iontype0-in-A`

(-1) Requested (=reference) number of ions of type 0 in compartment A. The default value of -1 means: use the number of ions as found in time step 0 as reference value.

`iontype0-in-B`

(-1) Reference number of ions of type 0 for compartment B.

`bulk-offsetA`

(0.0) Offset of the first swap layer from the compartment A midplane. By default (i.e. bulk offset = 0.0), ion/water exchanges happen between layers at maximum distance (= bulk concentration) to the split group layers. However, an offset b (-1.0 < b < +1.0) can be specified to offset the bulk layer from the middle at 0.0 towards one of the compartment-partitioning layers (at +/- 1.0).

`bulk-offsetB`

(0.0) Offset of the other swap layer from the compartment B midplane.

`threshold`

(1) Only swap ions if threshold difference to requested count is reached.

`cyl0-r`

(2.0) [nm] Radius of the split cylinder #0. Two split cylinders (mimicking the channel pores) can optionally be defined relative to the center of the split group. With the help of these cylinders it can be counted which ions have passed which channel. The split cylinder definition has no impact on whether or not ion/water swaps are done.

`cyl0-up`

(1.0) [nm] Upper extension of the split cylinder #0.

`cyl0-down`

(1.0) [nm] Lower extension of the split cylinder #0.

`cyl1-r`

(2.0) [nm] Radius of the split cylinder #1.

`cyl1-up`

(1.0) [nm] Upper extension of the split cylinder #1.

`cyl1-down`

(1.0) [nm] Lower extension of the split cylinder #1.

User defined thingies

`user1-grps`

`user2-grps`

`userint1 (0)`

`userint2 (0)`

`userint3 (0)`

`userint4 (0)`

`userreal1 (0)`

`userreal2 (0)`

`userreal3 (0)`

`userreal4 (0)`

These you can use if you modify code. You can pass integers and reals and groups to your subroutine. Check the inputrec definition in `src/gromacs/mdtypes/inputrec.h`

Removed features

This feature has been removed from GROMACS, but so that old mdp and tpr files cannot be mistakenly misused, we still parse this option. gmx grompp and gmx mdrun will issue a fatal error if this is set.

`adress`

(no)