7.3.16 Simulated annealing

Simulated annealing is controlled separately for each temperature group in GROMACS. The reference temperature is a piecewise linear function, but you can use an arbitrary number of points for each group, and choose either a single sequence or a periodic behaviour for each group. The actual annealing is performed by dynamically changing the reference temperature used in the thermostat algorithm selected, so remember that the system will usually not instantaneously reach the reference temperature!

annealing:
Type of annealing for each temperature group

no
No simulated annealing - just couple to reference temperature value.

single
A single sequence of annealing points. If your simulation is longer than the time of the last point, the temperature will be coupled to this constant value after the annealing sequence has reached the last time point.

periodic
The annealing will start over at the first reference point once the last reference time is reached. This is repeated until the simulation ends.

annealing-npoints:
A list with the number of annealing reference/control points used for each temperature group. Use 0 for groups that are not annealed. The number of entries should equal the number of temperature groups.

annealing-time:
List of times at the annealing reference/control points for each group. If you are using periodic annealing, the times will be used modulo the last value, *i.e.* if the values are 0, 5, 10, and 15, the coupling will restart at the 0ps value after 15ps, 30ps, 45ps, etc. The number of entries should equal the sum of the numbers given in `annealing-npoints`.

annealing-temp:
List of temperatures at the annealing reference/control points for each group. The number of entries should equal the sum of the numbers given in `annealing-npoints`.

Confused? OK, let's use an example. Assume you have two temperature groups, set the group selections to `annealing = single periodic`, the number of points of each

group to `annealing-npoints = 3 4`, the times to `annealing-time = 0 3 6 0 2 4 6` and finally temperatures to `annealing-temp = 298 280 270 298 320 320 298`. The first group will be coupled to 298K at 0ps, but the reference temperature will drop linearly to reach 280K at 3ps, and then linearly between 280K and 270K from 3ps to 6ps. After this is stays constant, at 270K. The second group is coupled to 298K at 0ps, it increases linearly to 320K at 2ps, where it stays constant until 4ps. Between 4ps and 6ps it decreases to 298K, and then it starts over with the same pattern again, *i.e.* rising linearly from 298K to 320K between 6ps and 8ps. Check the summary printed by `grompp` if you are unsure!

7.3.17 Velocity generation

`gen-vel:`

`no`
Do not generate velocities. The velocities are set to zero when there are no velocities in the input structure file.

`yes`
Generate velocities in `grompp` according to a Maxwell distribution at temperature `gen-temp` [K], with random seed `gen-seed`. This is only meaningful with integrator `md`.

`gen-temp: (300) [K]`
temperature for Maxwell distribution

`gen-seed: (-1) [integer]`
used to initialize random generator for random velocities, when `gen-seed` is set to -1, a pseudo random seed is used.

7.3.18 Bonds

`constraints:`

`none`
No constraints except for those defined explicitly in the topology, *i.e.* bonds are represented by a harmonic (or other) potential or a Morse potential (depending on the setting of `morse`) and angles by a harmonic (or other) potential.

`h-bonds`
Convert the bonds with H-atoms to constraints.

`all-bonds`
Convert all bonds to constraints.

`h-angles`
Convert all bonds and additionally the angles that involve H-atoms to bond-constraints.

`all-angles`
Convert all bonds and angles to bond-constraints.

constraint-algorithm:

LINCS
LINear Constraint Solver. With domain decomposition the parallel version P-LINCS is used. The accuracy in set with `lincs-order`, which sets the number of matrices in the expansion for the matrix inversion. After the matrix inversion correction the algorithm does an iterative correction to compensate for lengthening due to rotation. The number of such iterations can be controlled with `lincs-iter`. The root mean square relative constraint deviation is printed to the log file every `nstlog` steps. If a bond rotates more than `lincs-warnangle` [degrees] in one step, a warning will be printed both to the log file and to `stderr`. LINCS should not be used with coupled angle constraints.

SHAKE
SHAKE is slightly slower and less stable than LINCS, but does work with angle constraints. The relative tolerance is set with `shake-tol`, 0.0001 is a good value for "normal" MD. SHAKE does not support constraints between atoms on different nodes, thus it can not be used with domain decompositon when inter charge-group constraints are present. SHAKE can not be used with energy minimization.

continuation:
This option was formerly known as `unconstrained-start`.

no
apply constraints to the start configuration and reset shells

yes
do not apply constraints to the start configuration and do not reset shells, useful for exact coninuation and reruns

shake-tol: (0.0001)
relative tolerance for SHAKE

lincs-order: (4)
Highest order in the expansion of the constraint coupling matrix. When constraints form triangles, an additional expansion of the same order is applied on top of the normal expansion only for the couplings within such triangles. For "normal" MD simulations an order of 4 usually suffices, 6 is needed for large time-steps with virtual sites or BD. For accurate energy minimization an order of 8 or more might be required. With domain decomposition, the cell size is limited by the distance spanned by `lincs-order`+1 constraints. When one wants to scale further than this limit, one can decrease `lincs-order` and increase `lincs-iter`, since the accuracy does not deteriorate when (1+`lincs-iter`)*`lincs-order` remains constant.

lincs-iter: (1)
Number of iterations to correct for rotational lengthening in LINCS. For normal runs a single step is sufficient, but for NVE runs where you want to conserve energy accurately or for accurate energy minimization you might want to increase it to 2.

lincs-warnangle: (30) [degrees]
maximum angle that a bond can rotate before LINCS will complain

morse:

no
bonds are represented by a harmonic potential

yes
bonds are represented by a Morse potential

7.3.19 Energy group exclusions

energygrp-excl:
Pairs of energy groups for which all non-bonded interactions are excluded. An example: if you have two energy groups `Protein` and `SOL`, specifying

```
energygrp-excl = Protein Protein  SOL SOL
```

would give only the non-bonded interactions between the protein and the solvent. This is especially useful for speeding up energy calculations with `mdrun -rerun` and for excluding interactions within frozen groups.

7.3.20 Walls

nwall: 0
When set to `1` there is a wall at `z=0`, when set to `2` there is also a wall at `z=z-box`. Walls can only be used with `pbc=xy`. When set to `2` pressure coupling and Ewald summation can be used (it is usually best to use semiisotropic pressure coupling with the `x/y` compressibility set to 0, as otherwise the surface area will change). Walls interact wit the rest of the system through an optional `wall-atomtype`. Energy groups `wall0` and `wall1` (for `nwall=2`) are added automatically to monitor the interaction of energy groups with each wall. The center of mass motion removal will be turned off in the `z`-direction.

wall-atomtype:
the atom type name in the force field for each wall. By (for example) defining a special wall atom type in the topology with its own combination rules, this allows for independent tuning of the interaction of each atomtype with the walls.

wall-type:

9-3
LJ integrated over the volume behind the wall: 9-3 potential

10-4
LJ integrated over the wall surface: 10-4 potential

12-6
direct LJ potential with the z distance from the wall

table
user defined potentials indexed with the z distance from the wall, the tables are read

analogously to the `energygrp-table` option, where the first name is for a "normal" energy group and the second name is `wall0` or `wall1`, only the dispersion and repulsion columns are used

`wall-r-linpot: -1 (nm)`
Below this distance from the wall the potential is continued linearly and thus the force is constant. Setting this option to a postive value is especially useful for equilibration when some atoms are beyond a wall. When the value is ≤ 0 (<0 for `wall-type=table`), a fatal error is generated when atoms are beyond a wall.

`wall-density: [nm`$^{-3}$`/nm`$^{-2}$`]`
the number density of the atoms for each wall for wall types `9-3` and `10-4`

`wall-ewald-zfac: 3`
The scaling factor for the third box vector for Ewald summation only, the minimum is 2. Ewald summation can only be used with `nwall=2`, where one should use `ewald-geometry=3dc`. The empty layer in the box serves to decrease the unphysical Coulomb interaction between periodic images.

7.3.21 COM pulling

`pull:`

`no`
No center of mass pulling. All the following pull options will be ignored (and if present in the `.mdp` file, they unfortunately generate warnings)

`umbrella`
Center of mass pulling using an umbrella potential between the reference group and one or more groups.

`constraint`
Center of mass pulling using a constraint between the reference group and one or more groups. The setup is identical to the option `umbrella`, except for the fact that a rigid constraint is applied instead of a harmonic potential.

`constant-force`
Center of mass pulling using a linear potential and therefore a constant force. For this option there is no reference position and therefore the parameters `pull-init` and `pull-rate` are not used.

`pull-geometry:`

`distance`
Pull along the vector connecting the two groups. Components can be selected with `pull-dim`.

`direction`
Pull in the direction of `pull-vec`.

direction-periodic
As `direction`, but allows the distance to be larger than half the box size. With this geometry the box should not be dynamic (*e.g.* no pressure scaling) in the pull dimensions and the pull force is not added to virial.

cylinder
Designed for pulling with respect to a layer where the reference COM is given by a local cylindrical part of the reference group. The pulling is in the direction of `pull-vec`. From the reference group a cylinder is selected around the axis going through the pull group with direction `pull-vec` using two radii. The radius `pull-r1` gives the radius within which all the relative weights are one, between `pull-r1` and `pull-r0` the weights are switched to zero. Mass weighting is also used. Note that the radii should be smaller than half the box size. For tilted cylinders they should be even smaller than half the box size since the distance of an atom in the reference group from the COM of the pull group has both a radial and an axial component.

pull-dim: (Y Y Y)
the distance components to be used with geometry `distance`, and also sets which components are printed to the output files

pull-r1: (1) [nm]
the inner radius of the cylinder for geometry `cylinder`

pull-r0: (1) [nm]
the outer radius of the cylinder for geometry `cylinder`

pull-constr-tol: (1e-6)
the relative constraint tolerance for constraint pulling

pull-start:

no
do not modify `pull-init`

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

pull-print-reference: (10)

no
do not print the COM of the first group in each pull coordinate

yes
print the COM of the first group in each pull coordinate

pull-nstxout: (10)
frequency for writing out the COMs of all the pull group

pull-nstfout: (1)
frequency for writing out the force of all the pulled group

pull-ngroups: (1)
The number of pull groups, not including the absolute reference group, when used. Pull groups can be reused in multiple pull coordinates. Below only the pull options for group 1 are given, further groups simply increase the group index number.

pull-ncoords: (1)
The number of pull coordinates. Below only the pull options for coordinate 1 are given, further coordinates simply increase the coordinate index number.

pull-group1-name:
The name of the pull group, is looked up in the index file or in the default groups to obtain the atoms involved.

pull-group1-weights:
Optional relative weights which are multiplied with the masses of the atoms to give the total weight for the COM. The number should be 0, meaning all 1, or the number of atoms in the pull group.

pull-group1-pbcatom: (0)
The reference atom for the treatment of periodic boundary conditions inside the group (this has no effect on the treatment of the pbc between groups). This option is only important when the diameter of the pull group is larger than half the shortest box vector. For determining the COM, all atoms in the group are put at their periodic image which is closest to `pull-group1-pbcatom`. A value of 0 means that the middle atom (number wise) is used. This parameter is not used with geometry `cylinder`. A value of -1 turns on cosine weighting, which is useful for a group of molecules in a periodic system, *e.g.* a water slab (see Engin et al. J. Chem. Phys. B 2010).

pull-coord1-groups:
The two groups indices should be given on which this pull coordinate will operate. 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-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. `grompp` normalizes the vector.

pull-coord1-init: (0.0) [nm]
The reference distance at t=0.

pull-coord1-rate: (0) [nm/ps]
The rate of change of the reference position.

pull-coord1-k: (0) [kJ mol^{-1} nm^{-2} / [kJ mol^{-1} nm^{-1}]]
The force constant. For umbrella pulling this is the harmonic force constant in [kJ mol^{-1} nm^{-2}]. For constant force pulling this is the force constant of the linear potential, and thus minus (!) the constant force in [kJ mol^{-1} nm^{-1}].

pull-coord1-kB: (pull-k1) [kJ mol^{-1} nm^{-2} / [kJ mol^{-1} nm^{-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`.

7.3.22 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`;s upply `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 (default)
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 r^{-7} weighting of the atom pairs. 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)

7.3.23 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 `g_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:` ()
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 `g_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:` ()
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:` ()
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:` ()
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:` ()
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:` ()
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: ()
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 g_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.

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 `g_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)

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 `g_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 `g_energy` or used directly with `g_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.

7.3.24 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. Min1,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: Min1,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 reset as delta -¿ 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)

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 the value specified by `weight-equil-wl-delta`.

number-all-lambda
Expanded ensemble weight updating is stopped when the number of samples at all of the lambda states is greater than the value specified by `weight-equil-number-all-lambda`.

number-steps
Expanded ensemble weight updating is stopped when the number of steps is greater than the level specified by `weight-equil-number-steps`.

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 `weight-equil-number-samples`.

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 the value specified by `weight-equil-count-ratio`.

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)
Low temperature for simulated tempering.

sim-temp-high: (300)
High temperature for simulated tempering.

simulated-tempering-scaling: (linear)
Controls the way that the temperatures at intermediate lambdas are calculated from the `temperature-lambda` part of the lambda vector.

linear
Linearly interpolates the temperatures using the values of `temperature-lambda`,*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 ith state has temperature `sim-temp-low` + (`sim-temp-high-sim-temp-low`)*((exp(`temperat`
1)/(exp(1.0)-1)).

7.3.25 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.

7.3.26 Electric fields

E-x ; E-y ; E-z:
If you want to use an electric field in a direction, enter 3 numbers after the appropriate `E-*`, the first number: the number of cosines, only 1 is implemented (with frequency 0) so enter 1, the second number: the strength of the electric field in V nm^{-1}, the third number: the phase of the cosine, you can enter any number here since a cosine of frequency zero has no phase.

E-xt; E-yt; E-zt:
not implemented yet

¡h3¿Mixed quantum/classical molecular dynamics¡!–QuietIdx¿QM/MM¡!–EQuietIdx–¿¡/h3¿

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 ¡i¿ab initio¡/i¿ 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.

7.3.27 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 (1); gb-obc-beta (0.8); gb-obc-gamma (4.85);
Scale factors for the OBC model. Default values are 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 (default)

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 `grompp` using values that are specific for the choice of radii algorithm (0.0049 kcal/mol/Angstrom^2 for Still, 0.0054 kcal/mol/Angstrom^2 for HCT/OBC) Setting it to 0 will while using an sa-algorithm other than None means no non-polar calculations are done.

7.3.28 Adaptive Resolution Simulation

adress: (no)
Decide whether the AdResS feature is turned on.

adress-type: (Off)

Off
Do an AdResS simulation with weight equal 1, which is equivalent to an explicit (normal) MD simulation. The difference to disabled AdResS is that the AdResS variables are still read-in and hence are defined.

Constant
Do an AdResS simulation with a constant weight, `adress-const-wf` defines the value of the weight

XSplit
Do an AdResS simulation with simulation box split in x-direction, so basically the weight is only a function of the x coordinate and all distances are measured using the x coordinate only.

Sphere
Do an AdResS simulation with spherical explicit zone.

adress-const-wf: (1)
Provides the weight for a constant weight simulation (`adress-type`=Constant)

adress-ex-width: (0)
Width of the explicit zone, measured from `adress-reference-coords`.

adress-hy-width: (0)
Width of the hybrid zone.

adress-reference-coords: (0,0,0)
Position of the center of the explicit zone. Periodic boundary conditions apply for measuring the distance from it.

adress-cg-grp-names
The names of the coarse-grained energy groups. All other energy groups are considered explicit and their interactions will be automatically excluded with the coarse-grained groups.

adress-site: (COM) The mapping point from which the weight is calculated.

COM
The weight is calculated from the center of mass of each charge group.

COG
The weight is calculated from the center of geometry of each charge group.

Atom
The weight is calculated from the position of 1st atom of each charge group.

AtomPerAtom
The weight is calculated from the position of each individual atom.

adress-interface-correction: (Off)

Off
Do not a apply any interface correction.

thermoforce
Apply thermodynamic force interface correction. The table can be specified using the `-tabletf` option of `mdrun`. The table should contain the potential and force (acting on molecules) as function of the distance from `adress-reference-coords`.

adress-tf-grp-names
The names of the energy groups to which the `thermoforce` is applied if enabled in `adress-interface-correction`. If no group is given the default table is applied.

adress-ex-forcecap: (0)
Cap the force in the hybrid region, useful for big molecules. 0 disables force capping.

7.3.29 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 to your subroutine. Check the inputrec definition in `src/include/types/inputrec.h`