MC^{3}S

Monte Carlo for Complex Chemical Systems

Manual, Version 2.1

January 10$^{\text{th}}$, 2008

JMS & NR

Comments or questions: rai@chem.umn.edu

1 New features in version 2.1

- Automatic assignment of Ewald parameters and rcutchg, if requested
- Thermodynamic integration
- Expanded ensemble
- Addition of external electric field
- Calculation of solubility parameter, cohesive energy density, and heat of vaporization
- Ability to use atom displacement
- rigid swaps for fully flexible molecules
- Torsion potential using tabulated torsional potential
- Modifying the energy subroutines so that it doesn't use the neutral charge groups for Ewald summation
- Ability to not use Coulombic part in boltzmann weights
- Ability to use different pressures in different boxes, input pressure in MPa instead of simulation units
- Ability to use ghost ideal particles in the simulation, a number of particle offset, to prevent unnecessary memory usage
- Better way of excluding and including intra-molecular interactions
- Bug fixes
- System charge neutrality check at an appropriate place
- Extensive checks for the correctness of the fortran.4 input files for the molecule description part. Points the user where to look for a possible error while specifying LJ, stretch, bend, and torsion

2 Notes on Ewald summation

kalp should be atleast $3.2/rcutchg$. $Kamxl, Kmaxm$, and $Kmaxn$ would be set by the code to an acceptable value for most of the cases. In the event you suspect, that above criteria might not suit you well, then increase *kalp* and, maybe, increase $Kmaxl, Kmaxm$, and $Kmaxn$, manually in order to get a convergence in both parts of the Ewald summation.

3 Notes

This manual is intended to clarify the variables necessary for simulations carried out in the canonical (NVT), isobaric-isothermal (NpT), canonical Gibbs and isobaric-isothermal Gibbs ensembles. Simulations in the grand canonical (μVT) ensemble are also possible with this code but not discussed. Additionally, the Lennard-Jones 12-6 potential is assumed for non-bonded interactions. Other functional forms, such as exponential-6, 9-6 or mmff, are implemented but should be considered non-generalized and only used with caution.

Many important variables are not set in the input file but rather are determined prior to compiling the code. Most of these, such as **lnpt** or **lgibbs** are located in the file **control.inc**. Changing any variable in that file (or any other **.inc** file) will require you to recompile the code before taking effect. One should carefully check if all the logical parameters in **control.inc** conform to system that one is simulating.

To compile the code- edit the file **Makefile** to fit your specific machine/compiler, then execute **make**. To completely rebuild from source: execute **make clean** followed by **make**.

4 Input File, fort.4

4.1 System Specification

seed: Initializes the random number generator with an integer seed value. If the seed stays the same, the program will always generate the same sequence of pseudo-random numbers (useful for debugging). Sometimes this is not desirable, in which case it should be set to any other integer. Should be normally set to 0.

L_add: Useful if you want to add some helium type particles to the system (true). Usually (false)

N_add: Number of particles that you want to add to system

N_box2add: Which box you want to add to the system

N_moltyp2add: Molecule type of particles

L_sub: Useful if you want to subtract some helium type particles to the system (true). Usually (false)

N_sub: Number of the particles that you want to subtract from the system

N_box2sub: Which box you want to sustract the particles from

N_moltyp2sub: Molecule type of the particle getting subtracted

lecho: Echos the data read in by **readdat.f** to standard output. Should be set to true.

lverbose: Determines whether verbose output of input variables is desired (true) or not (false).

l_Coul_CBMC: To use the Coulombic energy during boltzmann weight calculation (true). Normally (true)

Num_cell_a: Number of unit cells in **a** direction in the case of solid simulations

Num_cell_b: Number of unit cells in **b** direction in the case of solid simulations

Num_cell_c: Number of unit cells in **c** direction in the case of solid simulations

run_num: Run number for the simulation, used to consistently name the output file

suffix: Similar to the run_num, an identifier

nstep: Number of Monte Carlo cycles (steps) to run simulation for.

lstep: If true, **nstep** refers to MC steps; if false, **nstep** refers to cycles. Controls all statements referring to "cycles (steps)" and should be false unless debugging.

lpresim: Do a presimulation to generate the SAFE-CBMC probabilities. Must be on one particle, with **pmfix** set to 0.

iupdatefix: Specifies how often the SAFE-CBMC probabilities should be updated.

L_tor_table: For the torsional potential. If you want to use tabulated potentials instead of regular functional form,set it to (true)

L_spline: if **L_tor_table** is true, then set to true if spline interpolation is requested

L_linear: if **L_tor_table** is true, then set to true if linear interpolation is requested

nbox: The number of simulation boxes for the current setup. Each box needs **boxlx, boxly, boxlz, lsolid, lrect, kalp, and rcutchg** specified for it, see below for their meanings.

express: The external pressure for each box in an NpT or NpT Gibbs Ensemble simulation, in MPa.

ghost_particles: A number of particle offset for a particular box, nbox values, usually set to zero

L_Ewald_Auto: Automatic assignment of Ewald parameters. Preferred method. If requested it will generate kalp(ibox) and rcutchg(ibox) based on the system size.

temp: The temperature, in K.

fqtemp: The fluctuating charge temperature, in K, for simulations using ANES-MC and polarizable models.

Elect_field Specified electric field in z direction in Volts/Å

ianalyze: How often to call anaylsis subroutine-it will be called every **inanalyze** cycles. Could be a number smaller than **imv** (see below) in order to get better statistics.

nbin: Number of bins for the radial distribution function calculation. A value of 200 should be adequate for most purposes.

lrdf: Logical variable used to determine whether to calulate rdf (TRUE) or not (FALSE).

lintra: Logical variable used to determine whether to include intramolecular distances to include in rdf (TRUE) or not (FALSE).

lstretch: Logical variable used to determine whether to calculate bond stretch distribution (TRUE) or not (FALSE).

lgvst: Used to determine whether to calculate gauche vs. trans statistics (TRUE) or not (FALSE).

lete: Used to determine whether to calculate end to end distribution of flexible chains (True) or not (False). Fort.4 should be set up so that first bead corresponds to one end of chain and the last bead corresponds to the other end.

lrhoz: Used to determine whether to calculate Z profiles for the solid slab and systems like self assembled monolayers (TRUE) or not (FALSE). Please make sure that solid box (in case of solid vapor simulations) is at the center of simulation cell.

bin_width: specifies bin width for the end to end distribution as well as for density profile. A value of 0.2 should be adequate for most purposes.

iprint: How often to print out the simulation status- it will be printed out every **iprint** cycles (steps). Useful to determine how a simulation is going before it's finished. Useful value is for 10-20 outputs per simulation.

imv: How often to output the system configuration to the movie file, **fort.10**, in cycles (steps). As this is used for subsequent analysis, and reasonably (statistically) independent results are desired, there's not much use in setting this to less than 500.

iratio: How often the maximum translation and maximum rotation values are updated, in cycles (steps). Smaller values are ok for equilibration (say 250 - 500). For production run, should be set to a number greater than **nstep**.

iblock: How many cycles are to be independently averaged for system properties, like p, ρ, molefraction, etc. 1000 cycles per block (at least) is a good value. Useful to see if a simulation is equilibrated, as there will be a constant drift in the block averages otherwise (for example, energy steadily decreasing).

idiele: How often to calculate the dielectric constant, ϵ, of the system. For single component NpT simulations, also controls frequency of output of volume and energy statistics as output in **fort.14** through **fort.19**. Currently works for 1-box, rigid molecules only!

L_movie_xyz: Set it to true if you want a movie file in xyz format

boxlx, boxly, boxlz: Specifies the x, y and z dimensions of the simulation box. Only used for initialization, unless the simulation is in the canonical (NVT) ensemble in which case these specify the current box lengths.

lsolid: Specifies whether to allow the x, y and z box lengths to vary independently (true) or not (false). Somewhat of a misnomer as this is also used for liquid slab simulations.

lrect: If **lsolid** is true, specifies whether the cell should be stay rectangular, with axes mutually perpendicular (true) or not (false). Has no effect if **lsolid** is false. Note that in the case of non-cubic, non-rectangular, you must provide a restart file (**fort.77**), that is, you can't set **linit** to true.

kalp: Ewald sum convergence parameter. If all charge interactions are being calculated (**lchgall** is true) then this should be set to 5.6 or greater. If a charge cutoff is being used

(**lchgall** is false) then the product of **kalp** and the cutoff should be 3.2 or greater. It is highly recommended that convergence of the Ewald sum is verified for your particular application.

rcutchg: Distance cutoff for charge-charge interactions, beyond which they don't get calculated. Has no meaning if **lchgall** is true and all charge-charge interactions are considered. Otherwise, a value below 12-14 Å is *not* recommended. Should generally be equal to **rcut** if used, see below.

nchain: Total number of molecules in the simulation.

nmolty: Number of different molecule types in the simulation.

moltyp: Number of molecules of each type, must be **nmolty** values specified.

lmixlb: Specifies that Lorentz-Berthelot combining rules should be used for the Lennard-Jones potential. True for the TraPPE force field, and what is used by default.

lmixjo: Specifies that the Jorgensen combining rules should be used for the Lennard-Jones potential. True for the OPLS force field.

nijspecial: Specifies whether to adjust the combining rule by a scaling parameter. If 0 then no scale parameter is used and **ispec, jspec, aspec** and **bspec** are ignored. If 1 then the form of the scaling parameter is used as input by **ispec, jspec, aspec** and **bspec**. Only for special cases.

ispec, jspec: Specifies what atom type parameters (as listed in **suijtab.f**) should be modified by the scale factors. Normally unused.

aspec, bspec: Specifies what scale factors to use. **aspec** modifies the combined σ_{ij} value and bspec modifies the combined ϵ_{ij} value.

rmin: Minimum cutoff for atom-atom distances. Any move bringing atoms closer than this distance will be automatically rejected. Ideally this can be set to any value as long as $g(r)$ is zero up to that distance. In practice, between 1.2 and 1.4 Å is a good choice for production simulations. Can be less for equilibration, even down to 0 Å.

rcut: Maximum cutoff of the Lennard-Jones potential, beyond which interactions are not calculated. A value of 9 Å is ok for equilibration, with 12-14 Å production.

rcutnn: Nearest neighbor bead-bead cutoff for neighbor list. Not normally used.

softcut: Upper bound on numbers to take the (negative) exponential of. Rules out low probability events. 100.0 is a reasonable value.

rcutin: Specifies inner cutoff for dual cutoff CBMC. During CBMC growths, only intermolecular interactions out to this value are considered. Smaller values will require less computer time while larger values will have higher acceptance rates. Values between 5 and 9 Å are typically used, but one should check the efficiency for their particular application.

rbsmax, rbsmin: Specifies maximum and minimum radii of "in" region during an AVBMC move. Usually something like 5 and 3 Å, respectively, for an alcohol, though very molecule dependent.

4.2 Molecule Specification

Each molecule must have its own specification section, where the connectivity and potential parameters are specified.

4.2.1 Molecular Parameters

nunit: Number of atoms the molecule is made up of. (United atom beads, e.g. CH_x units, are counted as one unit)

nugrow: The maximum number of atoms (beads) to regrow with CBMC. Should be equal to **nunit** unless **explct.f** is used.

ncarbon: For the explicit hydrogen model, the number of carbon atoms (which should be given first in the atom list) present which allows for simplified setup of the exclusion table. Should be equal to **nunit** for all other models.

maxcbmc: Maximum number of beads that are allowed to grow with CBMC. Should be equal to **nugrow**.

iurot: Specifies which bead the rotation move should be around. 0 specifies the center of mass and should be used as default.

lelect: Specifies whether this molecule has any charge sites (true) or not (false).

lflucq: Specifies whether the charges on this molecule are allowed to fluctuate (true). Only for polarizable models.

lqtrans: For polarizable models, specifies whether to allow intermolecular charge transfers to occur with this molecule type (true) or not.

lexpand: Specifies whether this molecule should be treated with the expanded ensemble (true) or not. Should be false by default.

lavbmc1, lavbmc2, lavbmc3: Specifies which of the three types of AVBMC moves to do upon this molecule. Used only if intrabox swap is specified (see **pmswap**).

fqegp: Specifies an energy offset in the fluctuating charge coulombic energy. Only for polarizable models.

maxgrow: Maximum number of interior segments to consider in a SAFE-CBMC regrowth, see **pmfix** below.

lring: Specifies whether the molecule is a flexible ring that should be regrown with SAFE-CBMC during a swap move. If true, **iring** must be specified on the following line.

lrigid: Specifies whether this molecule should be treated as a rigid body (true) or not. Can also be partially rigid. If set to true then the next input line must be which beads should be grown from if the molecule is partially rigid, or 0 if entirely rigid.

lrig: Specifies whether rigid segments should be grown from non-rigid segments during SAFE-CBMC. False by default, and if true, additional variables must be specified after this line.

lsetup: Specifies whether to try to automatically set up the bead-bead potential types (true) or not. Experimental!

isolute: How often to write out this molecule's configuration to the **fort.11** file for later analysis. Identical in function to **imv** but only for this molecule type. Unless specifically needed, a value larger than **ncycles** is recommended.

eta: Specifies any additional potential energy to uniformly add to a molecule in a particular simulation box. Must be one value for each simulation box, and is usually 0.

lq14scale: Specifies if 1-4 charge-charge interactions needs to calculated. Usually false for TraPPE force field

qscale: If lq14scale is true then by what factor to scale the 1-4 charge charge interactions

growpoints: Only needed if **lrigid** is set to true. Specifies the number of sites in a partially rigid molecule that CBMC growths should be attempted from. If non-zero must be followed by the site unit numbers, one per line.

4.2.2 Bead Specification and Connectivity

Each bead in the molecule (from 1 to **nunit**) has to have the following variables set:

unit: The number of the bead.

ntype: Specifies the interaction parameters of this bead, as given in the file **suijtab.f**

leaderq: Specifies neutral charge groups. Should be set to the lowest unit number in a neutral charge group, or set to **unit** otherwise. Only used if **lchgall** is false.

vibration: Specifies the number of bonds this unit has. The first line is the quantity, followed by one line for each bond with two values which specify the unit number the bond is to and the bond type, as given by the **suvibe.f** file.

bending: Specifies the number of bond bending interactions this unit has. Like the above **vibration** variable, the first line specifies the quantity, then following lines (one for each bend) specify the two units the bend is with (the closest bonded one should be first) followed by the bond type as given in the **suvibe.f** file.

torsion: Specifies the number of torsional interactions this unit has. Syntax like **vibration** and **bending** with the first line the number of torsions, then following lines (one for each) specifying the three beads the interaction is with (in the order of decreasing proximity to this unit) and the type as given by **vtorso.f** and **suvibe.f**.

4.2.3 AVBMC Variables

If any of the AVBMC moves were set to true in the molecular parameters section, the following three variables must also be set.

pmbias: For an AVBMC move, specifies the probability with which an "out to in" move is attempted, with an "in to out" move attempted otherwise.

pmbsmt: For AVBMC 2 and 3 moves, the probability with which the target site is a particular molecule type. Must be one value present for each molecule type.

pmbias2: For an AVBMC 3 move, the probability a molecule is taken from the "out" region and moved to a target site, with a molecule taken from another cluster otherwise.

4.3 Simulation Box and Move Probability Setup

licell: Specifies whether to use a link cell list (true) or not. Should only be used for large systems ($N > 2000$). Currently only implemented for use on one box.

rintramax: Specifies the maximum distance between one end of a molecule and the other for use in the link cell list.

boxlink: Specifies which simulation box to treat with a link cell list.

rmtrax, rmtray, rmtraz: Initial maximum translational displacements for all molecules in the x, y and z directions, in Å. Only used if **linit** is true, otherwise read from **fort.77** file.

rmrotx, rmroty, rmrotz: Initial maximum rotational displacements for all molecule types around the x, y and z axes, in radians. Only used if **linit** is true, otherwise read from **fort.77** file.

tatra: Target acceptance ratio for translational moves. 0.5 is a good value. Changed every **iratio** cycles (steps).

tarot: Target acceptance ratio for rotational moves. 0.5 is a good value. Changed every **iratio** cycles (steps).

linit: Specifies whether to initalize a simulation (start from the beginning) or not. Should always be false except for the first run.

lnewtab: For zeolite calculation, specifies whether to make a new table of the potential function. Normally false.

lreadq: Specifies whether to read bead charge values from the restart file, **fort.77** (if true) or to take the values from **suijtab.f**. Used only for fluctuating charge models.

lbranch: Specifies whether the molecular structure can be grown with CBMC (false) or if it's to be read from **fort.78** (true) when initializing a simulation. One value for each molecule type needed, and usually false unless CBMC can't grow this molecule.

inich: The inital number of each molecule type that should be in this box. For each simulation box, the variables **inich, inix, iniy, iniz, inirot, inimax, zshift, dshift,** and **nchoiq** must be set. Only used for initialization.

inix, iniy, iniz: Integer number of how many molecules should be placed along the x, y and z directions in the initial lattice.

inirot: Only applies to linear molecules. Specifies the initial rotation to be used for each new atom along the z axis, influencing the torsional angle. If 0 then each atom is randomly

rotated. If >0 then each atom is uniformly rotated by **inirot** degrees. If <0 then atoms are alternatively rotated in the y direction by $\pm$ **inirot**.

inimix: For mixture simulations, specifies the order in which to select molecules for the initial lattice. 0 places molecules randomly, >0 places all molecules of each type in order, and < 0 places molecules in alternating order by type.

zshift: Specifies a uniform shift the z direction so that terminal beads have a certain z coordinate. Used for monolayer calculations, should be zero otherwise.

dshift: Specifies amount to stagger each row in the x direction in the initial lattice by, in Å. Optimal spacing can be achieved by setting this to half of the spacing between molecules, which is given by the box length (**boxlx**) divided by the number of molecules along that direction (**inix**).

nchoiq: Specifies the number of times to call **flucq.f** to optimize the fluctuating charge models in this box. For polarizable models only.

rmvol: Specifies the initial value of the volume displacement. For an NpT simulation a value of 10^3 is appropriate, whereas for NVT simulations 10^{-3} is a good choice, due to the difference in units for the two ensembles.

tavol: Specifies the target acceptance rate for volume moves. For equilibration a value of 0.4 might be better, for production 0.5.

iratv: Specifies how often in cycles to adjust the maximum volume displacement. Analogous to **iratio** for translation and rotation displacements. As mentioned for **iratio**, smaller values are ok for equilibration (say 250 - 500). Should be set to a number greater than **nstep**.

iratp: Specifies how often to calculate the pressure, in cycles. As calculating the pressure is somewhat computationally intense, this should generally be set to 5 or less.

rmflcq Specifies the maximum fluctuating charge displacement. For polarizable models only.

taflcq Specifies the target acceptance rate for fluctuating charge moves. For polarizable models only.

4.3.1 Monte Carlo Move Probabilities

The simulation will carry out **ncycles** number of cycles made up of the following move types: volume, swatch, swap, cbmc, fluctuating charge, expanded ensemble, translations and rotations, denoted by **pmvol, pmswat, pmswap, pmcb, pmflcq, pmexpc,** and **pmtra** below. As move probabilities are given cumulatively the probability for a rotation move is simply 1 **- pmtra**, for a translation move it's **pmtra - pmexpc** (or **pmtra - pmcb** if **pmflcq** and **p**mexpc are set to zero, as is recommended most of the time). Submove probabilities, such as which molecule to perform a translation move on, are also cumulative.

pmvol: Specifies the probability to perform a volume move. Can start around 0.001 during equilibration and should be adjusted until there is about one accepted volume move every 10 cycles.

pmvlmt: Probability to perform volume moves on each box. Must be one probability for each simulation box.

nvolb: For NVT Gibbs Ensemble, the number of pairs of boxes to perform volume exchange moves on.

pmvolb: The probability to perform a volume exchange on each pair of boxes in an NVT Gibbs Ensemble. Must be **nvolb** values.

box5, box6: Specifies the pair of boxes for volume exchange. Must be one line for each pair specified in **nvolb**. Not used if the simulation is in the Canonical, Isobaric-isothermal or NpT Gibbs Ensemble.

pmvolx, pmvoly: Only required if at least one of the simulation boxes is non-cubic, this sets the (cumulative) probability to do moves in the x and y directions, with the z probability being 1 - **pmvoly**.

pmswat Specifies the probability to perform a swatch (CBMC identity exchange) move.

nswaty Number of pairs of molecule types to perform swatch moves on.

nswatb Specifies molecule types for the pairs in the swatch move. Must be **nswaty** pairs listed.

pmsatc: Once a swatch move is being performed, this specifies the probability to perform it on a particular pair. Must be **n**swaty values.

nsampos: Specifies the number of beads that can be left in the same positions. **nsampos, ncut, splist, gswatc, nswtcb, pmswtcb,** and **box_pair** must be specified for each pair of molecule types given by **nswaty**.

ncut: Number of CBMC growth sites the structure given by **n**sampos has to grow back the whole molecule.

splist: Specifies the pair(s) of beads to be kept in the same position. Must be **nsampos:** pairs.

gswatc 2x(ifrom, iprev): For both molecule types, the bead to be grown from and the previous bead must be specified. These will almost always be beads listed in **splist**.

nswtcb: Number of different box pairs to attempt swatch moves between.

pmswtcb: Probabilities to pick a particular box pair for a swatch move. Must be **nswtcb** values.

box_pair: Specifies the box pairs that this swatch move should be attempted between. Note that this can also list the same box twice, in which case the move is performed inside a single box.

pmswap: Specifies the probability to perform a swap move, removing a molecule from one simulation box and trying to place it in another by growing it with CBMC. Ideally this should be altered to get one accepted swap move per cycle.

pmswmt: Once a swap move is being performed, this specifies the probability to choose a particular molecule type. Must be one value for each molecule type.

nswapb: The number of box pairs to perform a swap move on. **nswapb, pmswapb,** and **box1, box2** must be specified for each molecule type.

pmswapb: The probability to perform a swap move on a particular pair of boxes. Must be **nswapb** values.

box1, box2: Specifies the box pairs to perform a swap move between for this molecule type. If the box pair is the same box specified twice, this becomes a move with the AVBMC algorithm, provided one of **lavbmc1, lavbmc2,** or **lavbmc3** have been set to true.

pmcb: Specifies the probability to perform a CBMC move, regrowing a random segment of a molecule in a position near its present one. Usually set to about one third for flexible systems.

pmcbmt: Once a CBMC move is being performed, this specifies the probability to choose a particular molecule type. There must be one value for each molecule type.

pmall: For each molecule type, the probability that it should try to regrow itself in its entirety when a CBMC move is performed. Usually set to zero. Must be one value for each molecule type.

pmfix: Specifies the probability to perform a SAFE-CBMC move on a molecule type, once it has been chosen for a CBMC move. Usually zero unless the molecule is large or is a flexible ring.

pmflcq: Specifies the probability to perform a fluctuating charge move. For polarizable models only.

pmfqmt: Once a fluctuating charge move is being performed, this specifies the probability to carry it out on a particular molecule type. Must be one value for each molecule type.

pmexpc: Specifies the probability to perform an expanded-ensemble move which changes a molecule's interaction parameters.

pmeemt: Once an expanded ensemble move is being performed, this specifies the probability to carry it out on a particular molecule type. Must be one value for each molecule type.

pmexpc1: Probability for new expanded ensemble i.e. to change the identity of the molecule.

pm_atom_tra: Probability to do atom translations, used only in special cases. Usually 0.0d0

pmtra: Specifies the probability to perform a translation move, changing a molecule's position. Usually set for about one third of the cumulative probability.

pmtrmt: Once a translation move is being performed, this specifies the probability to carry it out on a particular molecule type. Must be one value for each molecule type.

pmromt: Once a rotation move is being performed, this specifies the probability to carry it out on a particular molecule type. Must be one value for each molecule type.

nchoi1: Specifies the number of randomly chosen positions to consider for the CBMC swap move during the trial insertion of the first bead. 10 is a reasonable number. More will go

slower, but increase the acceptance rate of the swap move. There must be one value of **nchoi1** for each molecule type.

nchoi: Specifies the number of randomly chosen positions to consider during all CBMC regrowths. A larger value takes longer but increases acceptance rates. 8 is a reasonable value. There must be one value of **nchoi** for each molecule type.

nchoir: Specifies the number of rotations to consider for a rigid molecule's swap move. Only used if **.lrigid.** is true for this molecule. There must be one value of **nchoir** for each molecule type.

nchoih: Specifies the number of explicit hydrogen states to consider during a CBMC regrowth, actually just changes the methyl group's rotation. Unused for other molecules. There must be one value of **nchoih** for each molecule type.

nchoitor: Specifies the number of torsional angles to consider during a CBMC regrowth. 100 is a good choice. Larger takes longer but increases acceptance rates. There must be one value of **nchoitor** for each molecule type.

nchbna, nchbnb: Specifies the number of bend angles to consider during a CBMC regrowth. 1000 is a good choice. **nchbnb** is only used for branched molecules. Each molecule type has to have both specified.

icbdir: Specifies whether a CBMC growth should only go in one direction. 0 is usually chosen, meaning no preference. 1 indicates growths are only considered for increasing unit numbers. Must be one value for each molecule type

icbsta: If a consistent starting point is desired for a CBMC move, **icbsta** will specify the unit number. For example, +5 indicates CBMC growths should always start at bead number 5. However, a value of -5 indicates that the growth should start at a random position between bead 5 and the highest numbered bead (useful when one end of the chain is rigid and tethered to a surface). Usually chosen to be 0, meaning no preferred starting point. If not set to 0, make sure that the absolute value of icbsta + nugrow is equal to nunit + 1 for consistency. There must be one value of **icbsta** for each molecule type.

exclusion: Specifies whether to exclude any intermolecular interactions or not. If not, 0 should be input, otherwise an integer value determines how many pairs of exclusions there are. For each value, there must be one additional line specifying the molecule type and atom and the molecule type and atom it should be excluded from interacting with.

internal 1-4: By default, LJ interactions are considered only if beads are seperated by four or more bonds and coulombic interactions are considered past three bonds with 1-4 interactions scaled by **q14scale**. Set this integer variable equal to the desired number of exceptions to this rule, 0 for no exceptions. If non-zero, then there must be that many following lines each giving the molecule type, the two bead numbers to change the rule for, +1 or -1 (for turning the interaction on or off, respectively), the factor to scale the LJ interaction by, and the factor to scale the coulombic interaction by.

oh 1-5 interaction: Specifies whether any additional repulsive interactions should be computed for hydroxyl hydrogen- intramolecular oxygen pairs separated by 4 bonds. First number is the quantity of interactions, followed by one line for each interaction giving the molecule

type, bead numbers and a15 value, where value 1 is for an ether oxygen and 2 is for an alcohol oxygen.

lucall: Specifies whether to independently calculate the chemical potential (true) or not. Usually taken to be false as the chemical potential is automatically determined by the swap move.

ucheck: If **lucall** is true, specifies whether or not to calculate the chemical potential for each molecule type, with 0 meaning not to and a value larger than 0 meaning to calculate it.

nvirial: If **lvirial** is true in **control.inc** then the second virial coefficient will be calculated for a molecule, and **nvirial** determines the number of configurations to consider at each step. This is a specialized simulation and not for the general case.

start, step: If the second virial coefficient is being calculated, these determine the starting center of mass distance and step size, in Å.

Thermodynamic integration in stages
(see Maginn's paper for details of method). In topmon, here are the changes. There is a new logical variable called lmipsw in control.inc. If true, it does the thermodynamic integration (either use NVT or use NPT with zero percant volume move - hence NVT). Will fail for any other ensemble. And, only checked for one component systems.

No change in fort.4 - the input file is unaffected by the thermodynamic integration part. However, using lmipsw requires one new file called fort.35. The file is as follows -
1. **lwell** - set to true if stages b and c are done, false in case of stage a. For a definition of stages, see Maginn's paper.
2. **awell(i,j)** - input the strength of the external well in K in a matrix notation (interaction of i molecule site with j lattice site).
3. **bwell** - parameter for gaussian well width - 0.5 might be a good number.
4. **lstagea**, **lstageb**, or **lstagec** - make one of them true.
5. **etais**, **lamdbais** - the final weak potential, and value of **lambda** for that stage (see the paper).
6. box length - specify the boxlength at the beginning of stagea and the end of stagec (full stages) in hmatrix notation if lsolid is true and lrect is false. If lsolid is false, use the box length of the two cubic boxes. Cannot mix cubic/noncubic boxes - best is to use hmatrix notation.
7. **iratipsw** - the number of cycles in which the integrand is updated. If lstageb is true, make sure that this is an integral multiple of iratp.
8. reduced coordinates of all the lattice sites.

New expanded ensemble
It does work for changing the identity of one molecule (e.g., from ideal to a full molecule). e.g., for growing a solute molecule in solvent in a one-box NPT or NVT ensemble. It may work for Gibbs (swapping that molecule in one of its identities to the other box and then growing it) - but not extensively tested recently. This part of the code requires one change to the input in fort.4 - a variable called pmexpc1 - the probabiliity of doing the new expanded ensemble move. Other than that, there is a variable called lexpee in control.inc. If this

logical variable is true, then expanded ensmble is done and another input file called fort.45 is required. Also there is a smax variable - maximum number of stages that intialize the arrays.

The input file is as follows -
1. **imolty** on which ee is performed - molecule type. For example imolty = 1 is for the solvent, imolty = 2 is for the solute particles already in the solution, and imolty = 3 is the solute particle that we will try to grow in the solution for this simulation.
2. **nmolty1**: number of actual types of molecules in the simulation. In the above example - it is 3 (1 solvent, 1 solute molecules in full form, and 1 solute molecule that will go through stages).
3. final state: **fmstate** - the final state of the growing molecule (when it is fully grown). This determines the number of molecule types that are needed in fort.4. In the above example, it is fmstate+2. Each fmstate stages have one molecule type each, and one for the solvent, and one for the already present solutes.
4. weights associated with each stage: fmstate values. Could start with all zero and then optimize it so that all stages are visited with reasonable frequency.
5. **sstate1** and **sstate2**: In case expanded Gibbs ensemble is performed, the interdiate stages when the growing molecule is transferred from one box (sstate1) to the other (sstate2). Should be consequitive numbers. In case NPT or NVT expanded ensemble - make these numbers greater than fmstate (need to adjust smax in control such that these numbers are less than smax).
6. **eeratio**: In case of expanded Gibbs ensemble, the probability of exchanging the tagged particle (that is undergoing ee) at its end stages (full molecules in either box) with an untagged solute particle. For example, if stage 1 is in box 1 and stage 10 in box 2 - and at both these stages the tagged molecule is full molecule (just in different boxes) - then there should be a way to make another solute molecule tagged (otherwise only one molecule will switch between boxes). In case of NPT or NVT EE, make eeratio to be less than 0.
7. **mstate**: the current stage of the tagged molecule.

Additions for dipole moment and electric field

dipole.dat For an NVT simulation with **ldielect** equal to true, dipole.dat contains the x, y, and z components of the dipole moment (μ_x, μ_y, and μ_z) in units of e Å. These quantities may be used to calculate the dielectric constant using the following equation:

$$\epsilon = 1 + \frac{1}{4\pi\epsilon_0}\frac{4\pi}{3VT}\left(\langle\mu^2\rangle - \langle\mu_x\rangle^2 - \langle\mu_y\rangle^2 - \langle\mu_z\rangle^2\right) \tag{1}$$

see Allen and Tildesley, page 161.

lelect_field must be set to true in order for the electric field calculation to be turned on.

In fort.4:

Elect_field: the electric field strength, in units of V/Å. The electric field is applied in the z-direction. (this appears in fort.4 after fqtemp)

In order to calculate the interaction with an electric field, a new function (exfield) is called in sumup, energy, and boltz. exfield.f calculates the interaction with an electric field,

given by:

$$\begin{aligned} U_{\text{field}} &= -\mu \cdot E \\ &= -q * r_z * E \end{aligned}$$

5 Input / Output Files and Their Contents

5.1 Input Files

fort.4 As detailed above, this is the control file for the simulation that determines what the system is made up of, how long to run it, and what kinds of moves to attempt.

fort.77 This is the restart file, and is necessary for all simulations except when initializing. It is made up of the number of cycles already run, the molecules' maximum translational and rotational displacements, the volume displacements, the box lengths, as well as the type and current location (simulation box) of each molecule, and finally the coordinates and charge of each atom in the system. For a continuing simulation, the final configuration file, **final-config**, can be copied to `fort.77` to pick up where the previous simulation left off.

input_struc.xyz Only necessary if `lbranch` is true for any molecule. If so, this is where the molecule's structure is specified.

fort.23 Only necessary if a SAFE-CBMC simulation is being performed, this file contains the probability histograms used in that algorithm. To adapt the probabilities, the final histograms are output in the file `fort.22` and should then be copied to `fort.23` to continue on.

fort.7 Only necessary for expanded ensemble calculations, this file holds the current values of the Lennard Jones parameters.

fort.61, fort.62, fort.63 For molecule builder, the vibration, bending and torsion libraries. Experimental!

Depreciated Files

fort.25 Only for use in zeolite calculations, this file stores the Lennard Jones force field.

fort.91 Only for use in zeolite calculations, this file stores the tabulated zeolite potential energy.

5.2 Output Files

The majority of the information about the simulation is output run1a.dat.

config.dat The final configuration of the system as well as the maximum displacements and box lengths are contained in this file, the restart file. To continue a simulation, this file should be copied to `fort.77`.

fort.10 The movie file which holds the configurations of the system as output every `iblock` cycles. Used to determine the radial distribution functions, $g(r)$ among other things.

fort.11 The `isolute` movie file output every `isolute` cycles. Redundant to `fort.10` when so special outputting is desired.

fort.12 Contains the box lengths, total energy and number of each molecule type for each box, output every cycle. Useful for many things.

fort.13 If there is a non-cubic, non-rectangular box in the simulation, this file will contain the angles between cell vectors, written out every cycle.

fort.14, fort.15, fort.16, fort.17, fort.18 If the simulation is of a single molecule type and **idiele** is less than **nstep**, then these 5 files will contain averages related to the dielectric constant. **fort.14** and **fort.15** will have

$$\frac{4\pi}{3Vk_{\mathrm{B}}T}\frac{1}{4\pi\epsilon_0}\langle\mathbf{M}^2\rangle \quad \text{and} \quad \frac{4\pi}{3Vk_{\mathrm{B}}T}\frac{1}{4\pi\epsilon_0}(\langle\mathbf{M}^2\rangle - \langle\mathbf{M}_x\rangle^2 - \langle\mathbf{M}_y\rangle^2 - \langle\mathbf{M}_z\rangle^2)$$

respectively, where $\mathbf{M}$ is the total dipole moment of the box and $\mathbf{M}_x$, $\mathbf{M}_y$, and $\mathbf{M}_z$ its x, y and z components. Both quantities are unitless. Note that Allen & Tildesley relate the dielectric constant of the system ϵ, to these quantities *plus* 1. `fort.16, fort.17`, and `fort.18` will contain $\langle\mathbf{M}_x\rangle$, $\langle\mathbf{M}_y\rangle$, and $\langle\mathbf{M}_z\rangle$, respectively, in units of e Å.

fort.14, fort.15, fort.16, fort.17, fort.18, fort.19 If the simulation is of a single molecule type, **idiele** is less than **nstep** and **lnpt** is true, then these 6 files will contain the following averages: $\langle V\rangle, \langle V^2\rangle, \langle U\rangle, \langle U^2\rangle, \langle V\rangle\langle U\rangle$, and $\langle VU\rangle - \langle V\rangle\langle U\rangle$ where V is the box volume and U is the total energy, for each box. This is in addition to the dipole moment information contained in **fort.14-18** (see above).

fort.22 The final SAFE-CBMC probability histograms are contained in this file, if the SAFE-CBMC algorithm is used. Can be used to start a simulation with the new histograms by copying to **fort.23**.

fort.31, fort.32, fort.33 If the AVBMC algorithm is used, these files will contain the cluster statistics and energy ratios, output every cycle.

save-config Like the **final-config** restart file but output every **nstep**/10 if **nstep** is greater than 100 MC cycles otherwise it is not written. Useful if a simulation ends unexpectedly so that progress up to that point can be kept, in which case it should be copied to the next input configuration file, **fort.77**.

end2end_box1, end2end_box2, end2end_box3 These files contain the end to end vector distribution for box 1, box 2 and box 3 respectively. If the box is not present in simulation then that particular file will be empty.

rhoz_box1, rhoz_box2, rhoz_box3 These files contain z density profile for the respective boxes.

comrhoz_box1, comrhoz_box2, comrhoz_box3 These files contain center of mass z density profile for the respective boxes.

beadrdf_box1, beadrdf_box2, beadrdf_box3 These files contain bead-bead readial distribution functions for the the respective boxes.

comrdf_box1, comrdf_box2, comrdf_box3 These files contain center of mass readial distribution functions for the the respective boxes.

beadnum_box1, beadnum_box2, beadnum_box3 These files contain bead-bead number integrals for the the respective boxes.

comnum_box1, comnum_box2, comnum_box3 These files contain center of mass number integrals for the the respective boxes.

bendang_dist_box1, bendang_dist_box2, bendang_dist_box3 These files contain bending angle distribution for the respective boxes.

tors_frac_box1, tors_frac_box2, tors_frac_box3 These files contain torsion fractions for the respective boxes.

torsprob_box1, torsprob_box2, torsprob_box3 These files contain torsion angle probabilty distribution vs number of defects per chain, for the respective boxes.

Gauchedefects_box1, Gauchedefects_box2, Gauchedefects_box3 These files contain fraction gauche defects vs torsion for the respective boxes.

pattern_box1, pattern_box2, pattern_box3 These files contain pattern of g+, trans, g- for the respective boxes. Transform first number into base 3 to get the pattern of gauche defects where -1=g+, 0 = trans, 1=g+.

decoder This file contains decoder information for the gauche defect pattern.

Note for begginers Some basic guidelines about file management. **Every time a new run is carried out all the simulation output files are overwritten, so in order to preserve then one should copy the old files to a file with a new name.**

Force field development If one is developing force field for a particular molecule then it is important to keep the output file for example
% topmon > prod
where prod is the output file . It is not necessary to keep all other fort.* output files but for **final-config** , **fort.77**, **fort.4** to start the new trial. If you believe that you have parameters that are close to the desired value then you should carry out each simulation in a separate directory say T1, T2 (trial 1, trial2) etc and keep all the files for future reference. It is also a good idea to have a README file that has some basic iformation as to what is there in that particular directory.

Application oriented simulations If the objective of the simulation is to to provide microscopic understanding of the system then you should **save all the files** of your simulation. You might want to create new directory for different simulation runs and copy the old files to files with names appended by date or any other way you want to identify them.

6 Example Simulation

6.1 Starting Out, Melting and Cooling

Let's consider the calculation of the vapor-liquid coexistence curve for methanol. To do this, we'll use the NVT Gibbs ensemble to determine saturated liquid and vapor densities for a range of subcritical temperatures. Determining the vapor pressures will also let us predict what the normal boiling point is.

After specifying the connectivity and force field parameters in a **fort.4** file, we want to run the simulation. Since we don't have a restart file (**fort.77**), we set **linit** to true in **fort.4**. In this case, the box lengths for both simulation boxes will be read from **fort.4**, and you should pick densities that correspond roughly to a liquid and a vapor. Place between 10 and 33% of the molecules in the vapor box. If we're simulating 300 molecules (**nchain** = 300) then put 50 in the vapor phase and use 29 and 38 Å as box lengths for the liquid and vapor phases, which correspond to 0.55 and 0.048 g cm^{-3}, respectively. Split up the attempted Monte Carlo move probabilities so that we perform *no* volume moves (**pmvol** = 0) and evenly divide the probability into CBMC, translation and rotation by setting **pmcb** = 0.33 and **pmtra** = 0.67 and leaving all other probabilities at zero. Since we want to "melt" this structure away from the lattice it was set up on, set the temperature high, say 10,000 K, and run for 2,000 cycles to create a disordered system. For this process, a potential cutoff of 9 Å is sufficient.

Once this is done, we don't want to initialize again, so set **linit** to false and copy **final-config** to **fort.77**. At this point, we want to "cool" the simulation, so we leave the cutoff and probabilities the same and change the temperature to around 90% of the critical temperature, say 475 K, and then run for at least 5,000 cycles. If you set **iblock** to 1,000, at the end of the output file you will see 5 values of the energy (among other things) averaged over 1,000 cycles each, and can get a feel for how much it's changing. It'll probably be fairly steady after 2-3,000 cycles.

6.2 Equilibration

Now that we've brought the system down to a temperature that we want to get an accurate value of the saturated densites for, the next step is equilibration. Since we want to equilibrate density and chemical potential, we need to have moves that will change these. They are the volume and swap moves, respectively. We also want a more accurate calculation of the potential energy, so set the cutoff (**rcut**) to 14 Å. Don't forget to copy **final-config** to **fort.77** for each new simulation.

Ideally we want one accepted volume move for every ten cycles, so we should set **pmvol** to 0.2 / $N_{\text{molecules}}$ as we will be going for a target acceptance ratio of 50% for volume moves (**tavol** = 0.5). Set **iratv** so that the maximum volume displacement is allowed to change, say once every 250 cycles. Now we want to allow some swap moves too, so set **pmswap** = 0.01 and run the simulation for 10,000 cycles. Like the volume move, we want one accepted every 10 cycles, but we have no target acceptance ratio, so we have to see by trial and error what value of **pmswap** is appropriate.

For the stated probabilities, here's a sample of the output file as it pertains to the swap move:

```
### Molecule swap ###

molecule typ =     1
between box 1 and 2 into box 1
```

```
        uattempts = 14021.0 attempts = 14021.0 accepted = 254.0
suc.growth % = 99.501 accepted % = 1.812
between box 1 and 2 into box 2
        uattempts = 13851.0 attempts = 13851.0 accepted = 275.0
suc.growth % =100.000 accepted % = 1.985
```

As we ran 10,000 cycles, that means we had $10000/(254+275) \approx 19$ cycles between swap moves. Since we want that to be close to 10, we can increase **pmswap** to $0.01 \times \frac{19}{10} = 0.019$ for the next simulation. Note that since there is an overall flux of molecules from the liquid to the vapor box, the simulation has not reached equilibrium.

6.3 Production

Once we have reached stable vapor and liquid phases in our simulation, it's time to start producing data to collect. At this point, we should know that we'll be getting one accepted volume and one accepted swap move every 10 cycles. We don't need to update the maximum displacements, so we can set **iratio** and **iratv** to **nstep**. In fact, you could set the maximum x, y and z translational displacements to the average of the three, same for rotational displacements, in the **fort.77** file, if you want.

We can start calculating the pressure more frequently now, but it's still a fairly expensive calculation, so set **iratp** to say 5, which will calculate the pressure every 5 cycles. You will see that the pressure of a liquid is noisy, very noisy, and the standard deviation can be larger than the actual value! So in a case like this (NVT Gibbs ensemble) we say that the pressure between the two boxes is equilibrated due to the volume move, and just read the pressure from the vapor box.

6.4 Simulations at Lower Temperatures

When we have a decent result for 475 K, we can start thinking about calculating another state point at a lower temperature. To map out the coexistence curve you might go down in 50 K increments to 275 K. You could go lower but low temperatures take longer to equilibrate.

To keep the files separate, make a new directory for each temperature, in this case 425 K. Copy in the **fort.4** and **fort.77** files, and change the temperature in **fort.4** to 425. We will again have to equilibrate our simulation before we start collecting data. This mainly means volume and swap moves, as the vapor (and to a lesser extent liquid) density generally has a strong temperature dependence, especially near the critical point. Also, if you consider the lever rule, the relative amount of molecules in the vapor and liquid boxes will change (this is the same phenomena, just expressed differently). This means if we hold our box sizes equal, when we lower the temperature molecules will leave the vapor box in favor of the liquid. Our number of accepted swap moves will also decrease, as it becomes harder to swap at lower temperatures, so that we should adjust **pmswap** to bring the number back up to one accepted move per 10 cycles.

To ensure that we keep around 50 molecules in the vapor phase, we will have to *rescale* the vapor box lengths, but by how much? The answer is to compare the current number of molecules in the vapor phase to our target amount. Say that after 10,000 cycles at 425 K we wind up with 12 molecules in the vapor phase. We have $\frac{50}{12} \approx 4$ times too few molecules, so we want to increase the volume by a factor of 4. Since $V = L^3$, we need to uniformly increase each box length (L) by $4^{1/3}$ or 1.587. With our desired box length known, we then need to edit the restart file, **fort.77** to reflect this. For this case (one molecule type, two boxes), the lengths of the two boxes are on the 9^{th} and 10^{th} lines of the **fort.77** file. Note that changing the values in **fort.4** will have no effect on the simulation! Now equilibrate again and we should have close to 50 molecules in the vapor phase. From here, you can then start a production run to collect data at 425 K, and repeat the process for 375 K, 325 K, and so on. At some point, the amount of accepted swaps will become so low that it won't be possible to have one per 10 cycles. Then you can increase the number of trial positions investigated by increasing **nchoi1** and **nchoi**, however at increased computational cost. At 275 K, we might have to live with, say, only one accepted per 20 cycles.