Monte Carlo Simulation

Monte Carlo simulation is the application of Monte Carlo methods to molecular simulation. The $n^{-1/2}$ convergence properties of the Monte Carlo methods give them great advantage over methodical techniques when applied to the very high-dimensional integrals encountered in statistical mechanics. Moreover, sampling the complex shapes of the important regions of these integrals requires the application of a Markov chain, which is almost always accomplished via the Metropolis scheme or some extension of it.

Monte Carlo simulation involves the repeated application of an elementary procedure. First, a trial configuration is generated by making a perturbation to the a configuration of molecules. For example, a randomly selected atom is moved by some small amount from its present position. Then the ratio of probabilities for the trial and original configurations is computed, and from this quantity a decision is made whether to accept or reject the trial. If the trial is accepted, the new configuration is taken as the next state in the Markov chain, otherwise the original configuration is taken as the next state in the chain. Averages are collected over the many configurations so generated, until the computer resources budgeted to the calculation expire.

The power of the Monte Carlo method comes in the great variety perturbations that can be made to generate a trial configuration. The changes need not be physically realistic. One can, for example, take a molecule and change its species identity, say, converting a methane molecule to a water molecule. Such a move would not be made if one were attempting to generate samples in the canonical ensemble, but it might be done in a simulation of the grand-canonical ensemble, where the number of molecules of all types must fluctuate during a simulation. Thus the statistical mechanical ensemble is the first determinant of the types of moves that are attempted in a Monte Carlo simulation. Trials must be attempted that permit the system to sample the relevant microstates of the ensemble.

Within the constraints of sampling a given ensemble, one can be very creative in devising trials that significantly enhance the statistical quality of the calculation. As we learned in our study of Markov processes, it is advantageous to keep the system moving, and to avoid repeatedly visiting the same states over and over. It is also good to get a wide sampling of all possible configurations, and not to localize the sampling to a small corner of configuration space. Without cleverly constructed trial moves, it can be very difficult to encourage the system to explore a wide range of configurations. Good trial moves will transport the system in a single step from one configuration to another that is significantly different. A lot of progress in Monte Carlo simulation over the past decade has been made through the development of trial moves that work well in sampling complex fluids.

Most Monte Carlo simulations employ several types of trial moves to generate new configurations. For example, one trial may displace an atom, while another trial may rotate an entire molecule. At each elementary MC step, a decision is first made regarding which type of trial will be attempted. To satisfy detailed balance, this decision must be made via a random selection. Clearly, if an atom displacement trial is always followed

by a molecule rotation trial, then the transition probability is zero for reversing either of these moves in the subsequent step. The same may be said about the selection of the atom or molecule that is moved: if it is done in sequence (a trial and acceptance decision is made for one atom after another, following some predetermined order), there is no chance of reversing any of the moves in a subsequent step. Perhaps even worse, this prescription gives transition probabilities that depend on the (recent) history of the process, so the chain is not Markov. If instead all decisions are made probabilistically, so that for example there is a ½ probability of making either an atom move or molecule rotation in each elementary step, then the process is indeed Markov and detailed balance can be satisfied. While it is true that it is not necessary to satisfy detailed balance to generate the correct limiting distribution (ref Deem), our preference is to not invite problems and to use the stochastic approach to trial selection.

In our molecular simulation API, the type of simulation performed (MC versus MD) is specified by the Integrator. We have a class, IntegratorMC, that conducts Monte Carlo simulations. As just discussed, it repeated selects a trial move with some fixed pre-determined probability, and it executes the trial. However, this class is very general in that it contains no details about what perturbation is attempted in the trial. This information is coded into a separate MCMove class. Different subclasses of MCMove attempt different types of trials. At the outset of the simulation, the types of trials to be attempted are specified by adding instances of the corresponding MCMove subclasses to the IntegratorMC. Each MCMove subclass has associated with it a "frequency" value that indicates how likely it is to be selected as the trial for any given MC step. These values are used as unnormalized probabilities for selecting a trial. For example, instances of the MCMoveAtom and MCRotateMolecule classes may each have a frequency of 1, which means that at any point in the simulation they are equally likely to be selected as the next trial. There may be a third type of move, say MCMoveMolecule, having a frequency set (for example) to 2, which means that it is twice as likely of being selected as either of the other moves. Then for any elementary MC step, the likelihood of selecting an atom displacement, a molecule rotation, or a molecule displacement is in the ratio of 1:1:2, or probabilities of ¼, ¼, and ½, respectively.

Here's the basic code that performs the MC trial move. This is a method in the class IntegratorMC, and it is called repeatedly by the run method defined in the Integrator superclass. The frequencyTotal field is computed as the sum of the unnormalized frequencies assigned to each MCMove. The MCMoves are arranged in a linked list, so they can be iterated by calling the nextMove method present in each.

```
//Method from class IntegratorMC
public void doStep() {
   //Select a trial at random
  int i = (int)(rand.nextDouble()*frequencyTotal);
  trialMove = firstMove;
  while((i-=trialMove.getFrequency()) >= 0) {
    trialMove = trialMove.nextMove();
  }
  //Perform the trial and decide acceptance
  trialMove.doTrial(firstPhase)
}
```

The following initialize method is called at the start of the simulation. It doesn't do much more than compute the sum of the unnormalized frequencies. Some moves have a

```
//Method from class IntegratorMC
public void initialize() {
        deployAgents();      //put an Agent of this integrator in each
        frequencyTotal = 0; //compute frequency sum
        for(MCMove m=firstMove; m!=null; m=m.nextMove()) {
            m.resetFrequency(firstPhase);
            frequencyTotal += m.getFrequency();
        }
    }
```

frequency that is given as a nominal frequency times the number of molecules in the phase (so for example a particle displacement trial is attempted N times while a volume trial is attempted once, on average). This conversion is done by the resetFrequency method of MCMove.

The add method of IntegratorMC is used to add a subclass of MCMove to the repertoire of trials performed in the simulation. It inserts the given move into the linked list of moves maintained by the integrator.

```
//Method from class IntegratorMC
public void add(MCMove move) {
        if(firstMove == null) {firstMove = move;}
        else {lastMove.setNextMove(move);}
        lastMove = move;
        move.parentIntegrator = this;
    }
```

We will present some examples of specific MCMove subclasses as we discuss a few of the types of trials that are commonly employed in a MC simulation.

Displacement trial

The most basic of the MC trials is the simple movement of a randomly selected atom from one position to another. The new position is normally selected uniformly within some cubic region centered on the current position. A decision is made to accept or reject the trial such that the probability of acceptance gives an overall transition probability to satisfy microscopic reversibility. We will consider in detail how this probability is constructed, as the reasoning is common to MC trials of any complexity.

The limiting probability distribution is dictated by the ensemble being sampled, but in almost all situations the acceptance probability is exactly that developed in the canonical ensemble, so we will assume that NVT is the governing ensemble. In this case, the limiting probability distribution is

$$\pi(\mathbf{r}^N)d\mathbf{r}^N = \frac{1}{Z_N} e^{-\beta U(\mathbf{r}^N)} d\mathbf{r}^N \tag{1.1}$$

We include the differential elements $d\mathbf{r}^N$ to render these as probabilities and not probability densities. The momentum contributions have been integrated out.

The transition probabilities for the Markov process are constructed to ensure that the sample of states converges to this limiting distribution. In the Metropolis scheme introduced in the last section, the transition probability is formed as the product of the underlying transition probability τ and the acceptance probability $\min(1,\chi)$, where χ is chosen to enforce the detailed balance condition. To determine χ we must analyze the trial process in detail to evaluate τ.

The trial consists of the movement of atom k from position $\mathbf{r}^{\mathbf{old}}$ to position $\mathbf{r}^{\mathbf{new}}$. The probability that the Markov step selects this trial move can be constructed as in

Event [reverse event]	Probability [reverse probability]
Select molecule k [select molecule k]	1/N [1/N]
Move to r^{new} [move back to r^{old}]	d**r**/v [d**r**/v]
Accept move [accept move]	$\min(1,\chi)$ $[\min(1,1/\chi)]$

Illustration 1. We assume that we have already decided to make a displacement trial, and that the likelihood of making this choice is unchanging, and in particular is independent of the configuration of molecules. Then the first step in making the move happen is to select the atom k; the probablity that this will occur is 1/N, where N is the number of atoms in the system. Next the atom must be moved to a position within d**r** of $\mathbf{r}^{new}$; this probability is d**r**/v, where v is the volume of the region upon which the new position is selected uniformly. The reverse move must also be analyzed now. To do this we consider the likelihood that the same trial move would bring the system from a state in which molecule k is at the position we have designated $\mathbf{r}^{new}$ to the position we call $\mathbf{r}^{old}$. In this case the sequence of steps follows just as for the forward move. In this manner we conclude that the underlying forward- and reverse-step transition probabilities are

$$\tau_{ij} = \frac{1}{N} \times \frac{d\mathbf{r}}{v}$$
$$\tau_{ji} = \frac{1}{N} \times \frac{1}{v}$$

As discussed in our earlier introduction to the Metropolis scheme, the detailed balance condition requires that the forward-move acceptance probability χ satisfy

$$\pi_i \tau_{ij} \chi = \pi_j \tau_{ji}$$

With the ensemble distribution given by (1.1) this becomes

$$\frac{e^{-\beta U^{old}} d\mathbf{r}^N}{Z_N} \frac{1}{N} \times \frac{1}{v} \times \chi = \frac{e^{-\beta U^{new}} d\mathbf{r}^N}{Z_N} \frac{1}{N} \times \frac{1}{v}$$

Upon cancellation of like terms, this reduces to

$$\chi = e^{-\beta(U^{new} - U^{old})}$$

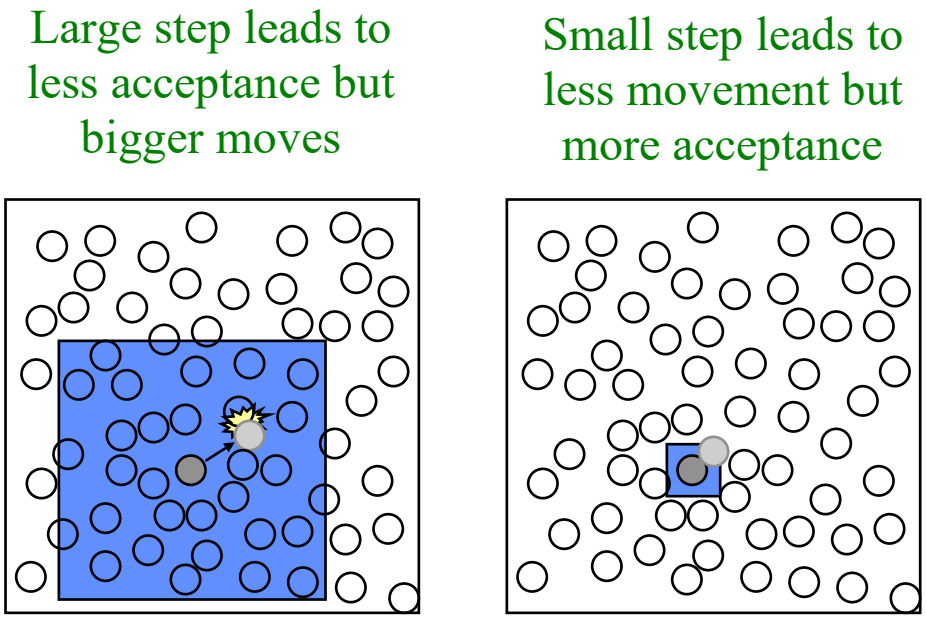
We note how fortunate it is that the configurational integrals Z_N cancel, as there is no simple means to determine them. Thus the acceptance can be completed knowing only unnormalized limiting-distribution probabilities.

The method in the MCMoveAtom class that completes this trial move follows. To have the MC simulation complete trials in which atoms are displaced, an instance of this class

```
//Method from class MCMoveAtom
public void thisTrial(Phase phase) {
        double uOld, uNew;
        if(phase.atomCount==0) {return;}                //no atoms to move

        int i = (int)(rand.nextDouble()*phase.atomCount); //pick a random number from 0 to N-1
        Atom a = phase.firstAtom();
        for(int j=i; --j>=0; ) {a = a.nextAtom();}        //get ith atom in list

        uOld = phase.potentialEnergy.currentValue(a);  //calculate its contribution to the energy
        a.displaceWithin(stepSize);                    //move it within a local volume
        phase.boundary().centralImage(a.coordinate.position()); //apply PBC
        uNew = phase.potentialEnergy.currentValue(a);  //calculate its new contribution to energy

        if(uNew < uOld) {                              //accept if energy decreased
            nAccept++;
            return;
        }
        if(uNew >= Double.MAX_VALUE || //reject if energy is huge or doesn't pass test
           Math.exp(-(uNew-uOld)/parentIntegrator.temperature) < rand.nextDouble()) {
             a.replace();              //...put it back in its original position
             return;
        }
        nAccept++;                     //if reached here, move is accepted
    }
```

must be added to the IntegratorMC class at the beginning of the simulation. Illustration 2 presents an applet that performs Monte Carlo simulation with atom displacement trials as given here.

An important performance consideration in conducting these trials involves the selection of the size of the step attempted in the displacement. Illustration 3 describes the issue. The trial consists of movement of the selected atom to a position selected uniformly in the blue region. It is desirable to have many accepted trials of very large atom displacements, but these goals are usually in conflict. A large displacement is likely to find the moved atom in a condition of overlap with another atom, with consequent rejection of the trial; small steps are likely to be accepted because they are accompanied by small energy changes. But a very small step doesn't do much to promote good sampling of configurations. An operational rule of thumb is to size the displacement region such that 50% of all trials are accepted. There is little quantitative analysis behind this choice, and there are arguments made for a much smaller target. This is particularly the case if the acceptance algorithm is alert to making the rejection decision before considering the total energy change; this can be done for example in hard-potential systems as soon as an overlap is discovered in the trial configuration.



The MCMove class has a method that performs tuning of parameters governing how the trial is conducted. It is shown here. Since it is defined and invoked in the superclass of all MCMove classes, it is inherited by them and thus does not have to be coded into each new MCMove subclass.

```
//Method from class MCMove
public void adjustStepSize() {
  if(nTrials == 0) {return;}
  if(nAccept > (int)(acceptanceTarget*nTrials)) {stepSize *= 1.05;}
  else{stepSize *= 0.95;}
  stepSize = Math.min(stepSize,stepSizeMax);
  stepSize = Math.max(stepSize,stepSizeMin);
  nTrialsSum += nTrials;
  nAcceptSum += nAccept;
  nTrials = 0;
  nAccept = 0;
}
```

Volume-change trial moves

In simulations performed in the NPT or other isobaric ensembles, it is required to conduct perturbations that lead the system to explore different volumes. An effective way to change the system volume is to complete the volume change while simultaneously scaling all the molecule center-of-mass positions (and thereby their mutual separations) by a linear scale consistent with the volume change. This idea was previously encountered by us when we derived the virial formula used to compute the pressure in a NVT simulation, and the reader should review that section if not familiar with it.

The limiting distribution in the NPT ensemble is

$$\pi\left(\mathbf{r}^N, V\right) = \frac{1}{\Delta_{\mathbf{r}}} e^{-\beta U\left(\mathbf{r}^N\right) - \beta PV} d\mathbf{r}^N dV$$

where again we ignore the momentum degrees of freedom, putting an **r** subscript on the partition function Δ to indicate that it does not contain the momentum contribution either. In terms of the scaled coordinates **s** = **r**/V introduced previously we have

$$\pi\left(\mathbf{s}^N, V\right) = \frac{1}{\Delta_{\mathbf{r}}} e^{-\beta U\left((V\mathbf{s})^N\right) - \beta PV} V^N d\mathbf{s}^N dV$$

In our simulation we sample $\mathbf{s}^N$ and V. Atom and molecule displacements are performed as described above; volume-change trials are conducted as described in Illustration 3. First a perturbation δ is selected uniformly with the range (-δV,+δV), and the new trial volume is computed as $V^{old} + \delta$. All molecule centers-of-mass are multiplied by the ratio of the new to the old linear dimensions of the system $r_i^{new} = r_i^{old} \times (V^{new} / V^{old})^{1/d}$, where *d* is the spatial dimension. In this manner the scaled coordinate **s** are held fixed during the transformation. The reverse move is completed by the same set of steps. From this specification we can easily write down the forward and reverse underlying transition probabilities for the Markov trial

$$\tau_{ij} = \frac{1}{2\delta V}$$

$$\tau_{ji} = \frac{1}{2\delta V}$$

Event [reverse event]	Probability [reverse probability]
Select V^{new} [select V^{old}]	$1/(2\,\delta V)$ [$1/(2\,\delta V)$]
Accept move [accept move]	$\mathrm{Min}(1,\chi)$ [$\mathrm{Min}(1,1/\chi)$]

With the Metropolis scheme for constructing transition probabilities, the correct limiting distribution is obtained if the acceptance probabilities are constructed to satisfy

$$\left[\frac{e^{-\beta(U^{old}+PV^{old})}\left(V^{old}\right)^{N}}{\Delta_{\mathbf{r}}}\right]\left[\frac{1}{2\delta V}\times\min(1,\chi)\right]=\left[\frac{e^{-\beta(U^{new}+PV^{new})}\left(V^{new}\right)^{N}}{\Delta_{\mathbf{r}}}\right]\left[\frac{1}{2\delta V}\times\min(1,\frac{1}{\chi})\right]$$

From which χ is

$$\chi=\exp\left[-\beta(\Delta U+P\Delta V)+N\ln(V^{new}/V^{old})\right]$$

where Δ indicates the difference (new)-(old). Thus a volume-change trial ($V^{\mathrm{old}}\rightarrow V^{new}$) performed as just described is accepted with probability $\min(1,\chi)$.

Volume-change trials are generally more expensive to conduct than a simple atom displacement, since each trial usually requires a full recalculation of the potential energy, with an $O(N^2)$ sum over all atom pairs (assuming no neighbor lists are used). In contrast an atom displacement requires an O(N) calculation involving a sum over all atoms interacting with the displaced one. On the other hand, a volume change represents a larger perturbation, and the degree that it changes the system is roughly comparable to performing a displacement trial once for each atom. Consequently, the frequency of volume-trial attempts is usually set to about 1/N times the frequency of a a single atom displacement trial. There can be mitigating factors. If the molecules are monatomic and the potential is spherically symmetric, it is usually possible to compute the energy by a simple rescaling of one or more atom sums, without conducting the $O(N^2)$ summation; these atom sums must be updated with each atom-displacement trial. If the potential is not pairwise additive, the atom-displacement trial may be much more expensive than an O(N) calculation.

Note that rescaling is performed to the molecule centers of mass, and that each atom position is not scaled separately. Intramolecular interactions are usually much more rigid than intermolecular ones, and by scaling only the centers of mass the volume trial does

not affect the intramolecular atomic distances. Doing otherwise—scaling the atom positions individually, and thereby their intramolecular separations—would greatly constrain the size of the volume perturbations that could be accomplished in a single trial.

As with the atom displacement trial, the maximum single-step volume change δV can be adjusted to improve the performance of the simulation. For a given N, we would expect that larger volumes should employ a larger δV for optimal performance (however that is defined). Normally we cannot let the step size depend on the volume, as this would introduce a bias toward smaller volumes, but through a simple reformulation of the probability distribution the same effect can be realized. In particular, if we consider lnV rather than V itself as the sampled quantity, we have for the limiting distribution

$$\pi\left(\mathbf{s}^N, \ln V\right) = \frac{1}{\Delta} e^{-\beta U\left((V\mathbf{s})^N\right) - \beta PV} V^{N+1} d\mathbf{s}^N d\ln V$$

In the trial move we take a step δ in lnV: $\left(\ln V\right)^{new} = \left(\ln V\right)^{old} + \delta$, with δ selected uniformly in some interval centered on zero, so the new and old volumes are related

$$V^{new} = V^{old} e^{\delta}$$

The trial is accepted with probability

$$\chi = \exp\left[-\beta(\Delta U + P\Delta V) + (N+1)\ln(V^{new}/V^{old})\right]$$

A Java code implementing this trial via a subclass of MCMove is shown here

```
// Method from class MCMoveVolume
public void thisTrial(Phase phase) {
  double hOld, hNew, vOld, vNew;
  vOld = phase.volume();
  hOld = phase.potentialEnergy.currentValue() //current value of the enthalpy
         + pressure*vOld;

  double vScale = (2.*rand.nextDouble()-1.)*stepSize; //choose step size
  vNew = vOld * Math.exp(vScale);              //Step in ln(V)
  double rScale = Math.exp(vScale/(double)Simulation.D); //evaluate linear scaling
  phase.inflate(rScale);                       //scale all center-of-mass coordinates

  hNew = phase.potentialEnergy.currentValue() //new value of the enthalpy
         + pressure*vNew;
  if(hNew >= Double.MAX_VALUE ||               //decide acceptance
     Math.exp(-(hNew-hOld)/parentIntegrator.temperature+(phase.moleculeCount+1)*vScale)
        < rand.nextDouble()) {
      phase.inflate(1.0/rScale);     //reject; put coordinates back to original positions
  }
  nAccept++;                         //accept
}
```

Illustration ? presents a NPT MC simulation applet.

Molecule insertion/deletion trials

We now turn to the last of the elementary trial moves that we will considerin in this section. The grand-canonical ensemble contains elements having differing numbers of molecules, so simulations of this ensemble must permit fluctuations in molecule numbers. The direct way to accomplish this is to attempt random insertions and deletions of molecules to and from the simulation volume. As before, these trials must be attempted and accepted to ensure that the simulation yields configurations according to the grand-canoncial probability distribution. Let us examine it now:

$$\pi\left(\mathbf{r}^N;N\right)=\frac{1}{\Xi}\frac{1}{\Lambda^{dN}}e^{-\beta U\left(\mathbf{r}^N\right)+\beta\mu N}d\mathbf{r}^N \tag{1.2}$$

Note that the momentum degrees of freedom depend on N and do not cancel with like terms in the partition function. It is convenient to introduce the absolute activity λ (which should not be confused with the activity defined in solution thermodynamics)

$$\lambda\equiv\frac{e^{\beta\mu}}{\Lambda^d}$$

which with Eq. (1.2) gives

$$\pi\left(\mathbf{r}^N;N\right)=\frac{1}{\Xi}\lambda^N e^{-\beta U\left(\mathbf{r}^N\right)}d\mathbf{r}^N \tag{1.3}$$

Note that we do not carry an N! term in this probability distribution. The term arises in the partition function only because of the phase space integration, which causes each configuration to be overrepresented N! times. The coordinates **r** in Eq. (1.3) represent positions in space—the interpretation is "this is the probability that we have *any molecule* within d**r** of the position $\mathbf{r}_1$, and *any other molecule* with d**r** of $\mathbf{r}_2$," and so on. Because the molecules are indistinguishable (assuming not a mixture), this specifies a microstate of the ensemble. In the phase-space integral the interpretation of **r** is different: each **r** there represents the coordinate of a particular molecule. Each microstate (in the sense of Eq. (1.3)) is represented N! times in the integral, once for each permutation of the positions of the molecules. Thus the phase-space integral must be accompanied by the N! division to render the correct partition function. We haven't had to pay much attention to this subtlety until now because N is unchanged by the trial moves we encountered so far.

As before we must consider both the forward and reverse trial moves to formulate the acceptance probabilities for each. The components of each trial and their probabilities are summarized in Illustration 4. We assume we have already made the decision that an insertion/deletion trial will be attempted. We then choose *with equal probability* whether to perform an insertion or a deletion trial. If an insertion we select a position $\mathbf{r}_{N+1}$

randomly with uniform probability over the entire simulation volume and the molecule is placed there. If a deletion we select a molecule randomly and remove it. The underlying transition probabilities specified by these moves are

$$\tau_{ij}(\text{insertion}) = \frac{1}{2} \times \frac{d\mathbf{r}}{V}$$

$$\tau_{ji}(\text{deletion}) = \frac{1}{2} \times \frac{1}{N+1}$$

Event [reverse event]	Probability [reverse probability]
Select insertion trial [select deletion trial]	½ [½]
Place molecule at r_{N+1} [delete molecule N+1]	dr/V [1/(N+1)]
Accept move [accept move]	min(1,χ) [min(1,1/χ)]

The detailed balance equation is as follows

$$\frac{e^{-\beta U^{old}} \lambda^N d\mathbf{r}^N}{\Xi} \left[\frac{1}{2} \times \frac{d\mathbf{r}}{V} \times \min(1, \chi) \right] = \frac{e^{-\beta U^{new}} \lambda^{N+1} d\mathbf{r}^{N+1}}{\Xi} \left[\frac{1}{2} \times \frac{1}{N+1} \times \min(1, \frac{1}{\chi}) \right]$$

solution for χ yields

$$\chi = \frac{\lambda V}{(N+1)} e^{-\beta \Delta U}$$

When one actually goes about performing the moves it is easy to be confused over the value of *N* as we use it here. To help clarify this, we write the acceptance probabilities specifically for the insertion and deletion moves. An insertion trial (N → N+1) is accepted with probability

$$A_{\text{insertion}} = \min\left[1, \frac{\lambda V}{N+1} e^{-(U_{N+1} - U_N)} \right]$$

while the deletion-trial (N → N-1) acceptance probability is

$$A_{\text{deletion}} = \min\left[1, \frac{N}{\lambda V} e^{-(U_{N-1}-U_N)}\right]$$

Note that insertions are favored by large values of λ and/or V, while the acceptance of a deletion trial is enhanced at large N. Importantly, note that when N = 0 the acceptance probability for a deletion trial is well defined, and is equal to zero. If the system samples a state having zero particles, there must still be a probability-½ attempt to delete a particle. This move will not be accepted, but doing otherwise—attempting only to insert when N = 0—leads to an inappropriate bias toward non-zero N. A rejected N = 0 deletion trial leads to another N = 0 configuration that rightly belongs in the simulation averages.

The following Java codes show how the insertion and deletion trials are implemented by the class MCMoveInsertDelete. The code here is appropriate for a single-component system (*i.e.*, it is not suitable for mixtures). First is the basic code that selects whether to attempt an insertion or a deletion.

```
//Method in class MCMoveInsertDelete
public final void thisTrial(Phase phase) {
    if(rand.nextDouble() < 0.5) {
      trialInsert(phase);
    }
    else {
      trialDelete(phase);
    }
}
```

The real work is done in the insertion/deletion methods.

```
//Method in MCMoveInsertDelete
private final void trialInsert(Phase phase, Species s) {
  Species.Agent s = phase.firstSpecies();
  double uNew;
  Molecule m = s.parentSpecies().getMolecule();  //make new molecule
  m.translateTo(phase.randomPosition());    //place it at random position
  uNew = phase.potentialEnergy.insertionValue(m); //compute molecule's energy
  if(uNew == Double.MAX_VALUE) {  //overlap
      return;
  }
   //here we take the absolute activity lambda = exp(+beta*mu)
  double bNew = Math.exp((mu-uNew)/parentIntegrator.temperature)
                * phase.volume()/(s.getNMolecules()+1);
  if(bNew > 1.0 || bNew > rand.nextDouble()) {  //accept
      phase.addMolecule(m);  //make trial molecule officially in phase
  }
```

```
//Method in MCMoveInsertDelete
private final void trialDelete(Phase phase) {
   Species.Agent s = phase.firstSpecies();
   if(s.getNMolecules() == 0) {return;}  //no molecules to delete; trial over
   double bOld, bNew;
   int i = (int)(rand.nextDouble()*s.getNMolecules()); //select a molecule
   Molecule m = s.firstMolecule;
   for(int j=i; --j>=0; ) {m = m.nextMolecule();}

   bOld = Math.exp((mu-phase.potentialEnergy.currentValue(m))
                                              /parentIntegrator.temperature);
   bNew = s.nMolecules/(phase.volume());
   if(bNew > bOld || bNew > rand.nextDouble()*bOld) {  //accept
       phase.deleteMolecule(m);
   }
}
```

A Java applet implementing a MC simulation in the grand-canonical ensemble may be viewed in Illustration ?.