



Molecular Monte Carlo Simulation II

Professor Richard J. Sadus

Centre for Molecular Simulation

Swinburne University of Technology

PO Box 218, Hawthorn Victoria 3122, Australia

Email: RSadus@swin.edu.au



Overview

This module comprises material for two lectures. The aim is to examine some of the theoretical basis of Monte Carlo simulations. This module builds on the concepts introduced in Module2.

The specific learning objectives are:

- (a) To understand Markov chains;
- (b) To understand Metropolis sampling;
- (c) To be able to write robust Monte Carlo programs in NVT ensemble.



Background

- In module 2, the concept of a Markov chain was introduced as a sequence of trials in which the outcome of successive trials depends only on the immediate predecessor.
- In a Markov chain, a new state will only be accepted if it is more ‘favourable’ than the existing state. In the context of a simulation using an ensemble, this usually means that the new trial state is lower in energy.
- A Markov chain is required to accurately determine the properties of the system in the finite time available for the simulation. The role of the Markov chain is to sample those states that make the most significant contributions.



Background (contd)

- A Monte Carlo simulation samples from a $3N$ -dimensional space represented by the positions of particles. Unlike molecular dynamics methods (Modules 7-12), particle momenta are not involved.
- It is not necessary to know particle momenta to calculate thermodynamic properties because the momenta contributes exclusively to the ideal gas term.
- Deviations from ideal gas behaviour are caused by interactions between particles which can be calculated from a potential energy function (Module 4).



Background (contd)

- The potential energy depends only on the positions of atoms and not their momenta.
- In effect, a Monte Carlo simulation calculates excess thermodynamic properties that result in deviations from ideal gas behaviour. The appropriate ideal gas term can be simply added at the conclusion of the simulation to obtain the total thermodynamic property.
- This Module builds on Module 2 by examining the basics of Monte Carlo Simulations and examining different ensemble.



Thermodynamic Averages

- Monte Carlo Simulation works by determining average thermodynamic properties.
- The average of any thermodynamic property $\langle A(\mathbf{r}^N) \rangle$ can be obtained by evaluating the following multidimensional integral over the $3N$ degrees of freedom on the N particles in the system.

$$\langle A(\mathbf{r}^N) \rangle = \int A(\mathbf{r}^N) \rho(\mathbf{r}^N) d\mathbf{r}^N$$

where $\rho(\mathbf{r}^N)$ is a probability distribution function (pdf) which reflects the probability of obtaining configuration \mathbf{r}^N which depends on the potential energy (E) of the configuration.



Thermodynamic Averages (contd)

$$\rho(\mathbf{r}^N) = \frac{\exp[-\beta E(\mathbf{r}^N)]}{\int \exp[-\beta E(\mathbf{r}^N)] d\mathbf{r}^N}$$

- These integrals cannot be evaluated analytically and conventional methods of integration are also not feasible. For example, to apply either Simpson's rule or the trapezium rule to evaluate a $3N$ -dimensional integral would require m^{3N} function evaluations, where m is the number of points required to determine the integral in each dimension.
- The Monte Carlo solution is to generate a large number of trial configurations \mathbf{r}^N and replace the integrations by summations over a finite number of configurations. If the configurations are chosen randomly, the thermodynamic average becomes:



Thermodynamic Averages (contd)

$$\langle A(\mathbf{r}^N) \rangle = \frac{\sum_{i=1}^{N_{trial}} A_i(\mathbf{r}^N) \exp[-\beta E_i(\mathbf{r}^N)]}{\sum_{i=1}^{N_{trial}} \exp[-\beta E_i(\mathbf{r}^N)]}$$

- In practice, this simple approach is not feasible because random sampling yields many configurations which have a very small Boltzmann factor.
- Such configurations make very little contribution to the average. Therefore, a prohibitively large number of configurations are required to obtain the correct answer.



Metropolis Sampling

- The limitations of random sampling can be avoided by generating configurations that make a large contribution to the thermodynamic average. This is the philosophy behind Metropolis sampling.
- Metropolis sampling biases the generation of configurations towards those that make the most significant contribution to the integral.
- It generates states with a probability of $\exp[-\beta E(\mathbf{r}^N)]$ and counts each of them equally.
- In contrast, simple Monte Carlo integration generates states with equal probability assigning them a weight of $\exp[-\beta E(\mathbf{r}^N)]$.



Metropolis Sampling (contd)

- Metropolis sampling generates a Markov chain which satisfies the conditions that:
 - (a) the outcome of each trial depends only on the preceding trail; and
 - (b) each trail belongs to a finite set of possible outcomes.

The former condition highlights the distinction between Monte Carlo and molecular dynamics simulation (Modules 7-12) methods. In a molecular dynamics simulation, all the states are connected in time.



Metropolis Sampling (contd)

- In 1953, Metropolis et al. showed that a transition probability matrix exists that ensures that the probability distribution function (pdf) is obeyed:

$$\left. \begin{aligned} \pi_{mn} &= C_{mn} & \frac{\rho_n}{\rho_m} \geq 1, m \neq n \\ \pi_{mn} &= C_{mn} \left(\frac{\rho_n}{\rho_m} \right) & \frac{\rho_n}{\rho_m} < 1, m \neq n \end{aligned} \right\}$$

where ρ_m and ρ_n are the probability densities for states m and n given by , and C_{mn} is the conditional probability of choosing n as the trial state. The calculation of the integral in the denominator of the pdf is not required because the above equation only involves the ratio of the probability densities.



Metropolis Sampling (contd)

Furthermore:

$$\sum_n \pi_{mn} = 1$$

$$\rho_m \pi_{mn} = \rho_n \pi_{nm}$$

- The Metropolis scheme is implemented by attempting a trial displacement and calculating the change in energy . If $\Delta E < 0$, then $\alpha > 1$ and the move is accepted. Otherwise, if $\Delta E > 0$, $\alpha < 1$ and the move is accepted with a probability of $\alpha = \exp(-\beta\Delta E)$. A random number R is generated and the move is accepted if $\exp(-\beta\Delta E) \geq R$.



Metropolis Sampling (contd)

- Typically, this procedure is summarised by defining the probability of acceptance as:

$$p = \min[1, \exp(-\beta\Delta E)]$$

- It should be noted that the above equation only represents the acceptance criteria resulting from molecular displacement. A different criterion is required as a result of either volume change or a change in a number of particles. You should recognise it as the acceptance criterion used in Module 2 for the NVT-ensemble.
- Lets now revisit the NVT-ensemble algorithm in greater detail because ALL other ensemble algorithms follow its basic approach.



Detailed NVT-Ensemble Algorithm

```
Part 1  loop cycle  $\rightarrow$  1 ... maxCycle
Part 2      Attempt trial displacements:
            loop atom  $\rightarrow$  1 ... maxAtom
Part 2.1      Generate a trial displacement:
                 $rx_{Trial} \leftarrow pbc(rx_{atom} + (2 \cdot rand() - 1) \times dMax)$ 
                 $ry_{Trial} \leftarrow pbc(ry_{atom} + (2 \cdot rand() - 1) \times dMax)$ 
                 $rz_{Trial} \leftarrow pbc(rz_{atom} + (2 \cdot rand() - 1) \times dMax)$ 
Part 2.2      Calculate change in energy:
                Calculate energy of the atom (ETrial) at trial position.
                 $\Delta E \leftarrow ETrial - E_{atom}$ 
Part 2.3      Apply Metropolis acceptance criterion:
                if ( $\Delta E < 0$  or  $\exp(-\beta \times \Delta E) \geq rand()$ )
                     $rx_{atom} \rightarrow rx_{Trial}$  //accept move
                     $ry_{atom} \rightarrow ry_{Trial}$ 
                     $rz_{atom} \rightarrow rz_{Trial}$ 
                     $E_{atom} \rightarrow ETrial$ 
                     $E \rightarrow E + \Delta E$ 
                     $numAccept \rightarrow numAccept + 1$ 
                end if
            end atom loop
```



Detailed NVT-Ensemble Algorithm (contd)

Part 2.4

Update periodically the maximum displacement:

```
if (mod(cycle, updateCycle) = 0)
    acceptRatio  $\leftarrow$  numAccept/(cycle  $\cdot$  atomMax)
    if (acceptRatio > 0.5)
        dMax  $\leftarrow$  1.05  $\times$  dMax
    else
        dMax  $\leftarrow$  0.95  $\times$  dMax
    end if
end if
end cycle loop
```



Detailed NVT-Ensemble Algorithm (contd)

- Typically, a Monte Carlo simulation is performed in cycles (Part 1). During each cycle, a displacement is attempted for every atom (Part 2). The atom to be displaced can be chosen randomly or alternatively, each atom can be considered in turn as illustrated above. It is also possible to displace every atom and apply the acceptance criterion to the combined move.
- New trial coordinates (Part 2.1) are obtained randomly up to a maximum value of $dMax$. This involves generating random numbers on the interval from 0 to 1 and applying periodic boundary conditions (pbc).
- The change in energy resulting from the trial displacement (Part 2.2) is simply the energy experienced by the atom at the trial position (E_{trial}) minus the energy of the atom at its existing position (E_{atom}).



Detailed NVT-Ensemble Algorithm (contd)

- This change in energy is evaluated and the Metropolis acceptance (Algorithm Part 2.3) rule is applied. If the move is accepted, the atom's coordinates, the atom's energy and the total energy (E) are updated.
- The maximum allowed displacement ($dMax$) affects the acceptance rate. Experience indicates that an acceptance rate of approximately 50 % is often desirable for a Monte Carlo simulation.
- A small value of $dMax$ will generally improve the acceptance rate but the phase space of the liquid may be sampled too slowly. Conversely, increasing the value of $dMax$ will reduce the acceptance rate. It is common to adjust the value of $dMax$ during the course of the simulation to maintain a 50 % acceptance rate.



Detailed NVT-Ensemble Algorithm (contd)

- Above (Part 2.4), the value of $dMax$ is adjusted at intervals specified by *updateCycle*. If $acceptRatio > 0.5$, the value of $dMax$ is increased, whereas it is decreased if $acceptRatio < 0.5$.
- It should be noted that there is no theoretical basis for using an acceptance rate of 50 % and in some cases it may be actually detrimental to efficient sampling.



Averages & Error Estimates

- The averages of a Monte Carlo simulation are only accumulated after an equilibration period. The number of cycles required for equilibrium is not known exactly beforehand and it should ideally be probed by test runs (e.g, by plotting density vs. cycle number).
- Alternatively, it is common to discard a large number of early runs to be on the safe side. For example, if the simulation runs for 20 000 cycles, the first 10 000 are discarded and averages are only accumulated for cycles 10 001 to 20 000.
- The average quoted is always the average of the entire post-equilibration period.



Averages and Errors (contd)

- It is always important to provide an error estimate of a simulation quantity. The error represents the deviation from the average value over the post-equilibrium cycles. If the error estimate is large it may indicate that equilibrium has not been reached or that the simulation is unreliable for some reason. The reasons for large errors include both errors in the code or an actual physical limitation, e.g., correlation-length induced fluctuations at the critical point.
- There are alternative ways of estimating errors. Some simulators average over independent runs commencing from different initial configurations. However, more commonly errors are estimated from a single run by dividing the post-equilibrium run into several blocks



Averages and Errors (contd)

The average of a property ($\langle A \rangle$) is determined from:

$$\langle A \rangle_{run} = \frac{1}{N - N_{equil} - 1} \sum_{i > N_{equil}}^N A_i$$

The post-equilibrium run is divided into N_b blocks of length l such that $lN_b = N$. The mean value of A is calculated for each block

$$\langle A \rangle_b = \frac{1}{l} \sum_{i=1}^{n_b} A_i$$

The standard deviation is then determined by averaging over all of the blocks:



Averages and Errors (contd)

$$s = \sqrt{\frac{1}{n_b} \sum_{b=1}^{n_b} (\langle A \rangle_b - \langle A \rangle_{run})^2}$$

This standard deviation that is commonly quoted as the error associated with a simulation quantity.

- It should be noted, that some authors quote the uncertainty as: $s / \sqrt{n_b}$ which will always be much less than the standard deviation. Therefore, some caution is required when comparing error estimates from different workers.



Project 1

Instead of a series of problems, it is timely to implement a ‘major’ programming project, namely using the detailed NVT-ensemble algorithm discussed in this chapter, develop your own working NVT MC program in C. The program should use the Lennard-Jones potential contain the following functions to:

- (a) initialise atoms on a fcc lattice;
- (b) calculate random numbers;
- (c) evaluate energy;
- (d) accumulate the average energy;
- (e) accumulate statistics;
- (f) print the output;

It should also use structures to contain related data such as the atomic coordinates.



Reading Material

The material covered in this module is discussed in greater detail in the following books:

M.P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, OUP, Oxford, 1987.

D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, Academic Press, San Diego, 1996, pages.

R.J. Sadus, *Molecular Simulation of Fluids: Theory, Algorithm and Object-Oriented*, Elsevier, Amsterdam, 1999.