MONTECARLOSIMULATIONS

Objectives

After completing the reading of this chapter, you will be able to:
Make use of random numbers in simulation.
Follow how a simple Monte Carlo scheme works.
Devise the Metropolis algorithm for simple systems.

Keywords

Random number, Monte Carlo method,
Evaluation of π, Metropolis Monte Carlo,
Probability of accepting or rejecting moves.

Introduction

In the earlier two chapters you have learnt how to construct a molecular dynamics (MD) simulation methodology for simple atomic/molecular systems. The generated trajectories by the MD technique are, in a sense, deterministic. This means, given a potential function of a system, one can choose a convenient time step, usually a few fs(femtoseconds), to generate the real time trajectories. But this time step is too small to probe real physical processes which call for very long MD simulations. When computation of an exact result by simulations appear infeasible, one can make use of the Monte Carlo methods.

The Monte Carlo methods

The Monte Carlo (MC) methods were developed during the fifties of the last century by von Neumann, Ulam and Metropolis. The name 'Monte Carlo' was chosen by Metropolis, because of the extensive use of the random numbers in the method. Prior to the development of the Monte Carlo methods, statisticians had used model sampling experiments to investigate complex problems. The model sampling experiments involve the generation of random numbers which were used in several arithmetic and logical operations. The use of computers have made these exercises easy and in the literature, one comes across many anecdotes narrating the efforts of the pioneers to make the Monte Carlo methods to be the most powerful and commonly used technique for analyzing complex problems.

Example of a Monte Carlo method: Evaluation of π

The use of the MC technique as a method of integration may be illustrated by the evaluation of the constantπ. For the purpose, we need to find the area of a circle of unit radius ($r = 1$). The circle is centred at the origin and is inscribed in a square (Figure 28.1). Thus the square has the length of each side as $2r$ (i.e. 2 units).

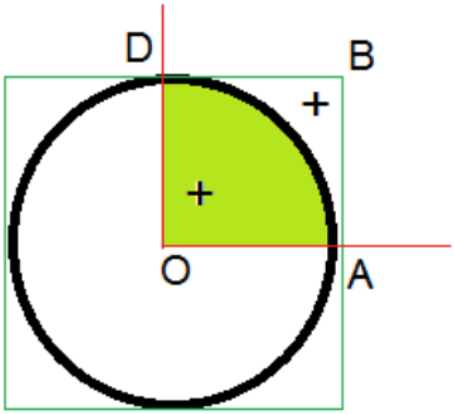

Figure 28.1: Area of a circle and hence the evaluation of π by the Monte Carlo method. Radius OA = r = 1 unit. The square in which the circle is inscribed has its side equal to $2r$ or 2 units.

From simple geometrical considerations, area of the circle = $\pi r^2 = \pi$sq units. Area of the square = $(2r)^2 = 4$ sq units. The ratio of the area of the circle to the area of the square is,

$$\text{Ratio} = \frac{\text{Area of the circle}}{\text{Area of the square}} = \frac{\pi}{4} \qquad (28.1)$$

Thus, if we can compute the ratio accurately, we should be able to evaluate π, by multiplying the ratio with 4. To do this, one can divide the whole square into smaller squares (say, 2^n squares, n very large) and count how many smaller squares fit into the area of the circle completely. Say, for example, we divide the square into $2^{10} = 1024$ smaller squares, out of which 812 small squares fit into the area of the circle, then the ratio = 812/1024 = 0.792969. This yields π = 3.171875.

In a typical Monte Carlo exercise, we need to compute the number of points that lies inside the circle out of a total number of points tried. The ratio of these two numbers equals $\pi/4$. The accuracy of the ratio depends on the total number of points used, with more points leading to a more accurate value of $\pi/4$. A number of trial shots are generated in the square OABD (Fig. 28.1) and at each trial, two independent random numbers are chosen from a uniform distribution on (0,1). These numbers are used as the coordinates of a point (the + marks in the figure 28.1). The distance of the random point from the origin O is calculated. If this distance is less than or equal to radius of the circle (*i.e.*, 1 unit), the shot has landed in the shaded region and is counted as a hit. Out of a total of τ_{shot} fired, if there are τ_{hit} hits counted, then

$$\frac{\tau_{\text{hit}}}{\tau_{\text{shot}}} = \frac{\pi}{4} \quad (28.2)$$

The success and accuracy of this method depend on the generation of $2\tau_{\text{shot}}$ random numbers from a uniform distribution. The larger the total number of shots used, more accurate value of the ratio $\pi/4$ is obtained.

A simple random number generator

In the simple Monte Carlo exercise discussed above, you have learnt how the random numbers are used to define the two coordinates, which subsequently lead to the evaluation of π. When the random numbers are chosen from a uniform distribution (typically between 0 and 1), these are called uniform deviates. Random numbers may also be drawn from a normal (Gaussian) distribution of specified mean and standard deviation. For the Monte Carlo simulations, it is always recommended to use the system-supplied random number generator. Usually, such a random number generator makes use of the time of the system clock. For example, in a HP Superdome system, the random number calling statements are:

```
c
c     invoking the system clock
callsystem_clock (icount(17))
callrandom_seed (put=icount)
c
c     calling the random number generator
c     (a library subroutine)
callrandom_number (harvest=xx)
callrandom_number (harvest=yy)
callrandom_number (harvest=zz)
c
```

Below we give a uniform random number generator program, which may be used for simple Monte Carlo exercises.

```
c
c     ************************
real function ran0(idum)
c     ************************
c     minimal random number generator of park and miller
c     ref: w. h. press, s. a. teukolsky, w. t. vetterling,
c     b. p. flannery;
c     numerical recipes in fortran, page 270
c     returns a uniform random deviate between 0.0 and 1.0.
c     set or reset idum to any integer value
cto initialize the sequence.
cidum must not be altered between calls for
c     successive deviates in a sequence.
c
integeridum,ia,im,iq,ir,mask
integer k
```

```
real am
parameter (ia=16807, im=2147483647, iq=127773, ir=2836)
parameter (mask=123459876)
am=1.0/float(im)
c
idum=ieor(idum,mask)
      k=idum/iq
idum=ia*(idum-k*iq)-ir*k
if (idum.lt.0) idum=idum+im
      ran0=am*float(idum)
idum=ieor(idum,mask)
c
return
end
c
```

The calling statements for using this function are:

```
do 110 j=1,nbead
c
c     using the random number generator ran0
c     to generate coordinates of a chain of 'nbead'
c
kx=(j)
ky=(j+1)
kz=(j+2)
c
xx=ran0(kx)
yy=ran0(ky)
zz=ran0(kz)
c
etc. etc.
c
  110 continue
c
```

MetroploisMonte Carlo method

In anysampling method, where the configurations for the system are chosen at random, a vast majority of the configurations will have energy very different from the average energy of the system. This results that the randomly chosen configurations will have insignificant contributions to the system energies. In order to avoid such situations, many improvement schemes (such as importance sampling, smart Monte Carlo, Metropolis Monte Carlo) have been suggested. Of these, the Metropolis Monte Carlo method is the most widely used and is discussed below.

The Metroplois Monte Carlo method is a computational approach for generating a set of *N*configurations,χ_1, χ_2, …, $\chi_N$of the system such that,

$$\lim_{N\to\infty}\left(\frac{N_\chi}{N}\right) = P(\chi) \qquad (28.3)$$

where, $P(\chi)$ is a given probability distribution and N_χ is the number of configurations χ.The probability distribution of the configurations is usually proportional the Boltzmann factor,$exp\left(\frac{-\Delta E}{k_B T}\right)$ where,ΔE is the difference in energy of the two configurations, k_B is the Boltzmann constant and T is the absolute temperature.

The central idea of the Metropolis method is to choose the representative configurations not completely at random, but in a manner that the selection is somehow 'biased' towards the configurations that have significant population at equilibrium. Let, the first (initial) configuration,$\chi_1$of the system is chosen at random. The subsequent configurations,$\chi_i$are then chosen with an appropriateprobability,$W(\chi_{i-1} \rightarrow \chi_i)$,from the previous configurations,χ_{i-1}. In order to make such a chain of configurations converge to a desired canonical distribution, we need to impose the condition of detailed balance;

$$P_{eq}(\chi_l).W(\chi_l \rightarrow \chi_m) = P_{eq}(\chi_m).W(\chi_m \rightarrow \chi_l) \qquad (28.4)$$

where, χ_l and χ_m are any arbitrary pair of configurations, $P_{eq}(\chi_l)$ and$P_{eq}(\chi_m)$ are the corresponding equilibrium probabilities;

$$P_{eq}(\chi_i) = Z^{-1} exp\left(\frac{-E(\chi_i)}{k_B T}\right) \qquad (28.5)$$

where, Z is the partition function and E is the total energy of the configuration χ_i.

The condition of detailed balance[eq. (28.4)]implies that, at equilibrium, the average number of moves,$(\chi_l \rightarrow \chi_m)$is the same as the average number of inverse moves, $(\chi_m \rightarrow \chi_l)$. This ensures the maintenance of equilibrium of the system during the moves. If the system was not in equilibrium initially, then the ratio of the probabilities of the two configurations tends to increase (if the initialvalue of the ratio was below its equilibrium value) or decrease (if the initialvalue of the ratio was above its equilibrium value). Therefore, it follows that for sufficiently long simulations the system will reach the state of thermodynamic equilibrium.

It is usually convenient to restrict the possible moves from a particularconfiguration only to a restricted number of 'adjacent' configurations. This is implemented in the following way.

1. If,$W(\chi_l \rightarrow \chi_m) = 0$, then, $W(\chi_m \rightarrow \chi_l) = 0$ (28.6)
2. If, $W(\chi_l \rightarrow \chi_m) \neq 0$, then

$$\frac{W(\chi_l \rightarrow \chi_m)}{W(\chi_m \rightarrow \chi_l)} = exp\left(\frac{-[E(\chi_m) - E(\chi_l)]}{k_B T}\right) = exp\left(\frac{-\Delta E}{k_B T}\right) \qquad (28.7)$$

This means that, a move from one configuration to another is possible, if and only if, the inverse move is also possible. In other words, two arbitrary configurations must be necessarily either mutually adjacent or not adjacent. If the two configurations are mutually adjacent then the probability of a move between them is related to the probability of inverse move by the well-defined relation, dependent only on the difference in energy between the two configurations.

We now write down the steps in the implementation of the Metropolis algorithm. (1) The first configuration is randomly generated. (2) At each point in the generation of the chain of configurations, a move is attempted from the current configuration. (3) The move is rejected immediately if the local chain configuration is not compatible with the attempted move. (4) If the difference between the energy of the resulting configuration and the energy of the current configuration, (ΔE), is negative (*i.e.*, the energy of the resulting configuration is less than the energy of the current configuration), then the resulting configuration is accepted and it becomes the new configuration in the chain. (5a) If$\Delta E$is positive, however, a (pseudo)random number R, between 0 and 1($0<R<1$), is generated and the resulting configuration is accepted, if, $exp\left(\frac{-\Delta E}{k_B T}\right) > R$. (5b) For, $exp\left(\frac{-\Delta E}{k_B T}\right) < R$, the resulting configuration is rejected. (6) Whenever the configuration resultingfrom the attempted move is rejected, the new configuration of the chain is the same as the current configuration.

The Metropolis criterion can then be summarized by the following expression for the probability of acceptance of a configuration of an attempted move:

$$P_{accept} = \min\left[1, exp\left(\frac{-\Delta E}{k_B T}\right)\right] \quad (28.8)$$

Note that the ratio between the acceptance probabilities of a move is related to the acceptance probability for the inverse move by the same relation [eq. (28.7)]. The transition probability between any two configurations is the product of the probability of attempting ($P_{attempt}$) a given adjacent configuration, and the probability of accepting (P_{accept}) the configuration. Then the condition of detailed balance [eq. (28.4)] is valid because probability of attempting ($P_{attempt}$) is constant. This is the reason why local moves that are not compatible with the chain configurationmust be considered, during the choice of configurations to be attempted in the simulation in theconfigurational space, even if they will get rejected.

Questions

1. In order to evaluate π, why the generated random numbers need to be between 0 and 1?
2. What happens if the sample size is just 2^4, in the evaluation of π exercise?
3. What is the importance of choosing the Boltzmann factor as a representative of probabilities?
4. How does the principle of detailed balance work?