



# Molecular Monte Carlo Simulation I

Professor Richard J. Sadus

*Centre for Molecular Simulation*

*Swinburne University of Technology*

*PO Box 218, Hawthorn Victoria 3122, Australia*

*Email: [RSadus@swin.edu.au](mailto:RSadus@swin.edu.au)*



# Overview

This module comprises material for two lectures. The aim is to introduce the ‘tools’ and basic concepts behind a Monte Carlo simulation.

The specific learning objectives are:

- (a) To understand how the potential energy is calculated;
- (b) To understand how to implement periodic boundaries and the minimum image convention;
- (c) To be able to write simple Monte Carlo programs in the NVT-ensemble.



## What is Molecular Monte Carlo Simulation?

- In Module 1, the Monte Carlo concept was introduced. We should how random numbers could be used to solve problems such as difficult multiple integrals. This relatively straight forward problem solving strategy is only one aspect of Monte Carlo. More generally, Monte Carlo can be used to *simulate* natural processes that are intractable by conventional means.
- All natural phenomena, such as the boiling point of a fluid, the viscosity or hardness of a material etc are caused by intermolecular interactions. Monte Carlo can be used to simulate these interactions.
- In some disciplines, the word ‘simulate’ is synonymous with ‘crude approximation.’ A ‘simulated’ solution is obtained because the exact solution is intractable.



## What is MMC Simulation? (contd)

- However, this connotation should not be applied to molecular Monte Carlo (MMC) simulation. In contrast to other theoretical approaches such as equations of state, perturbation theories, empirical correlations etc, MMC yields exact solutions based on the evaluation of intermolecular interactions.
- To illustrate the difference, consider an equation of state for hard spheres. A popular hard sphere equation is the Carnahan-Starling equation which can be used to determine the compressibility factor ( $Z$ ) of a fluid in terms of its packing fraction via the following relationship:

$$Z = \frac{1 + y + y^2 - y^3}{(1 - y)^3}$$



## What is MMC Simulation? (contd)

- The above model is quite accurate but we know exactly how hard spheres should interact. The potential ( $u$ ) between hard spheres of a particular diameter ( $\sigma$ ) is given by:

$$u(r) = \begin{cases} \infty & r \leq \mathbf{S} \\ 0 & r > \mathbf{S} \end{cases}$$

- A MMC simulation can be devised in strict accordance with the above potential. The resulting compressibility factors are exact because no extraneous approximation was used. Indeed, MMC data of hard spheres are used to test the accuracy of hard sphere equations of state such as the Carnahan-Starling equation. This applies to all other hard sphere theories.



## Generalised MMC Algorithm

- At the outset it is useful to have a generalised framework for the MMC algorithm. MMC simulation works by generating transitions between different states. This involves :
  - (a) generating a random trial configuration;
  - (b) evaluating an ‘acceptance criterion’ by calculating the change in energy and other properties in the trial configuration; and
  - (c) comparing the acceptance criterion to a random number and either accepting or rejecting the trial configuration



## Generalised MMC Algorithm (contd)

A general MMC Algorithm is illustrated below:

Part 1 Create an initial configuration (*config*).

Part 2 Generate a Markov Chain for *Ncycles*:

**loop**  $i \leftarrow 1 \dots NCycles$

    Create a new configuration (*configTrial*).

    Find the transition probability  $w(\text{config}, \text{configTrial})$ .

    Generate a uniform random number (*R*) between 0 and 1.

**if** ( $w > R$ ) **then**

        Accept move ( $\text{config} = \text{configTrial}$ ).

**else**

        Reject move (*config* is unaltered).

**end if**

**end loop**



## Generalised MMC Algorithm (contd)

- After an initial configuration is generated (Part 1), the algorithm works (Part 2) by repeatedly either accepting or rejecting new configurations. The algorithm introduces the concept of a ‘Markov chain.’
- It is important to realise that not all states will make a significant contribution to the configurational properties of the system. We must confine our attention to those states that make the most significant contributions in order to accurately determine the properties of the system in a finite time. This is achieved via a Markov chain (see later modules) which is a sequence of trails for which the outcome of successive trails depend only on the immediate predecessor.
- In a Markov chain, a new state will only be accepted if it is more ‘favorable than the existing state.’



## Generalised MMC Algorithm (contd)

- At this stage there are two important facets of the above algorithm:
  - (i) the evaluation of configurations involves assumptions concerning the nature and extent of intermolecular interactions; and
  - (ii) the acceptance criteria is ensemble-dependent.
- The rest of this Module will address issue (i) whereas the ensemble-dependency we be deferred to Module 3. In any programming examples to follow, it will be NVT ensemble will be assumed implicitly.



## Generalised MMC Algorithm (contd)

- In the NVT-ensemble, the only MC move is molecular displacement which is governed by the following acceptance criteria:

$$W(i) = \exp(-E(i) / kT)$$

where  $E$  is the configurational or potential energy of state  $i$ ,  $k$  is Boltzmann's constant and  $T$  is the temperature.



## Contributions to the Potential Energy

- At the outset it is important to realise that, in contrast to molecular dynamics, the kinetic energy has no role to play in an MMC simulation. The acceptance criterion only involves ‘configurational’ properties. Therefore, we are only interested in calculating the configurational of potential energy of the system.
- In general, the potential energy ( $E_{pot}$ ) of  $N$  interacting particles can be evaluated as

$$E_{pot} = \sum_i u_1(r_i) + \sum_i \sum_{j>i} u_2(r_i, r_j) + \sum_i \sum_{j>i} \sum_{k>j>i} u_3(r_i, r_j, r_k) + \dots$$

- where the first term represents the effect of an external field and the remaining terms represent particle interactions, i.e.,  $u_2$  is the potential between pairs of particles and  $u_3$  is the potential between particle triplets etc.



## Contributions to the Potential Energy (contd)

- Typically, it is assumed that only two-body interactions are important and the above equation is truncated after the second term.
- Two-body interactions undoubtedly make the most significant contribution to particle interactions, however, there is evidence that three-body interactions may be important in some cases.
- Because the evaluation of particle interaction is the most time-consuming step, the simulation algorithm is of order  $N^m$ , where  $m \geq 2$  denotes the number of interactions. Consequently, including three- or more-body interactions imposes a very large increase in computing time compared with two-body calculations.



## Contributions to the Potential Energy (contd)

- To use the above relationship we implicitly make two important assumptions:
  - (a) We assume that the interactions of the fluid are limited to pair interactions.
  - (b) We assume some form of the intermolecular potential ( $u$ ) between pairs of atoms.
- For the case of the hard-sphere potential, discussed above the calculation of pair interactions is straight forward (either zero or infinity). However, most realistic potentials are continuous.
- The rest of this module is concerned with the mechanics of calculating intermolecular interactions and intermolecular potentials are discussed at length in Module 3.



## Contributions to the Potential Energy (contd)

- For the time being, an example of a widely used (to the point of being over used) potential is the Lennard-Jones potential which contains two parameters that reflect the depth of the potential well ( $\epsilon$ ) and molecular diameter ( $\sigma$ ):

$$u(r) = 4\epsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right]$$

- As discussed in Module 3, this is a ‘convenient potential’ rather than being a genuine representation of pair interaction. However, the form of the potential is at least qualitatively captures the change of energy with intermolecular separation.



## Reduced Units

- Irrespective of the complexity of the intermolecular potential, it invariably involves at least two parameters namely,  $\epsilon$  and  $\sigma$ . It is very common to perform simulations in dimensionless units relative to these parameters. Examples include:

density	$\rho^* = \rho\sigma^3$
temperature	$T^* = kT/\epsilon$
energy	$E^* = E/\epsilon$
pressure	$P^* = P\sigma^3/\epsilon$
time	$t^* = t(\epsilon/m\sigma^2)^{0.5}$
force	$f^* = f\sigma/\epsilon$



## Short-Range vs. Long-Range Interactions

- Molecular forces can be characterised as either short-range or long-range and different techniques are required to simulate both types of properties. A long-range force is defined as one which falls off no faster than  $r^{-d}$  where  $d$  is the dimensionality of the system. Typically, ion-ion and dipole-dipole potentials are proportional to  $r^{-1}$  and  $r^{-3}$ , respectively. Dispersion and repulsion are examples of short-range forces.
- Many computation-saving devices can be employed to calculate short-range interactions such as periodic boundary conditions and neighbour lists. However, calculating long-range interaction requires special methods (see Module #) such as the Ewald sum, reaction field, and particle-mesh methods, because the effect of long-range interaction extends well past half the length of the simulation box. Here we will concentrate exclusively on short-range forces.



## Naïve Energy Calculation

- If intermolecular interaction is restricted to pairs of  $N$  molecules, the total potential energy ( $E_{pot}$ ) is obtained by summing over the contribution all of the individual pairs.

$$E_{pot} = \sum_{i>j}^{N-1} \sum_{j=1}^N u(r_{ij})$$

- Notice that indices of the above summation are deliberately chosen to avoid “self” interactions. Obviously, these “self” interactions do not contribute to the potential energy of the system and time should not be wasted calculating them.



## Naïve Energy Calculation (contd)

- The calculation of energy can be achieved by using the following nested double loop:

```
loop  $i \rightarrow 1 \dots N - 1$   
  loop  $j \rightarrow i + 1 \dots N$   
    Evaluate  $r_{ij}$ .  
    Evaluate  $u(r_{ij})$ .  
    Accumulate energy.  
  end  $j$  loop  
end  $i$  loop
```



## Naïve Energy Calculation (contd)

- The above pseudo code follows directly from the problem description and illustrates the nesting of loops required for the evaluation of either forces or energy.
- Care is taken to avoid identical molecules and a total of  $N(N-1)/2$  iterations will be performed.
- This algorithm is used generally with a very important modification.
- Simulations are performed typically with a few hundred molecules arranged commonly on a cubic lattice. The small sample size means that a large fraction of the molecules can be expected at the surface of the lattice rather than in the bulk. This is undesirable because the forces experienced by surface molecules are different from the bulk. Periodic boundary conditions are used to avoid this problem.



## Periodic Boundary Conditions & Minimum Image Convention

- Periodic boundary conditions (PBC) can be applied by constructing an infinite lattice that replicates the cubic simulation box throughout space.
- Molecules in any box on this lattice have a mirror image counterpart in all the other boxes.
- Changes in one box are matched exactly in the other boxes. If a molecule leaves a box then its counterpart in a neighbouring box enters through the opposite face. Consequently, the central box is devoid of any boundaries, and surface effects are eliminated.



## PBC & MIC (contd)

- The use of a periodic boundary condition solves the problem of surface molecules but it poses another potential difficulty for the simulation.
- The heart of a simulation program is the calculation of either molecular forces or energies. If there are  $N$  molecules in the simulation, then to calculate the pairwise force or energy experienced by each other molecule involves a summation of  $N - 1$  terms.
- In principle, we are also faced with the impossible task of including the interactions of the infinite array of periodic images. In the case of a short-ranged intermolecular potential, this problem is avoided by invoking the minimum image convention (MIC).



## PBC & MIC (contd)

- To calculate the intermolecular interaction of any molecule, we position it at the centre of a box with dimensions identical to the simulation box. The central molecule interacts with all molecules whose centres fall within this region, i.e., the closest periodic images of the other  $N - 1$  molecules.
- If the origin of the coordinate system is taken as the centre of the simulation box of length  $L$ , then the algorithms for applying the minimum image convention and periodic boundaries are very simple. All the coordinates lie within the range of  $1/2 L$  and  $-1/2L$ . Consequently, when the molecule leaves the box, we can focus attention on its mirror image by either adding or subtracting  $L$  to its coordinates.



## PBC & MIC (contd)

- This can be easily coded in FORTRAN using the intrinsic `anint(x)` to determine the nearest integer of  $x$ . For example, to add the correct number of box lengths to the molecular coordinates  $r_{ij}$  or the pair separation vector, we write:

```
rxij(i) = rxij(i) - boxl * anint( rxij(i) / boxl )
```

- The C/C++ programming language does not have a nearest integer function as part of its standard maths library. Consequently, we must write our own function to determine the required number of multiples to add. An example of such a function (`nearestInt`) is. The corresponding C/C++ statement is:

```
rxij[i] = rxij[i] - boxl * nearestInt(rx[ij], boxl);
```



## PBC & MIC (contd)

- The above statements are usually evaluated in a loop, e.g., during the evaluation of energies. The procedure for calculating the energy using periodic boundary conditions is given below.

```
loop  $i \rightarrow 1 \dots N - 1$   
  loop  $j \rightarrow i + 1 \dots N$   
    Evaluate  $r_{ij}$ .  
    Convert  $r_{ij}$  to its periodic image ( $r'_{ij}$ ).  
    if ( $r'_i < \text{cutOffDistance}$ )  
      Evaluate  $u(r'_{ij})$ .  
      Accumulate energy.  
    end if  
  end j loop  
end i loop
```



## PBC & MIC (contd)

- The above algorithm is different from the naïve approach in two important respects.
  - (a) The accumulated energy and forces are calculated for the periodic separation distances. Secondly,
  - (b) Only molecules separated by a distance less than the cut-off distance contribute to the calculated energy or forces.
- The cut-off distance is a consequence of the periodic boundary conditions. It can be any distance up to half the length of the simulation box. A higher cut-off is not permitted because it would violate the minimum image convention.
- Consequently, the above algorithm evaluates the properties of a truncated intermolecular potential rather than the full potential.



## Long-Range Corrections to PBC

- When PBC are used, the full contributions of the intermolecular potential can be obtained by using long-range correction terms.
- The contribution to a property  $X$  of the fluid from the missing long-range part of the potential can be obtained by adding a long-range correction.

$$X_{full} = X_c + X_{lrc}$$

- The long-range corrections are calculated by assuming that the pair-distribution function is unity beyond the cut-off distance  $r_c$ . Using this simplification, the long-range corrections for energy ( $E$ ), the virial term ( $W$ ), and the chemical potential ( $\mu$ ), are obtained from:



## Long-Range Corrections to PBC (contd)

$$E_{lrc} = 2\pi N\rho \int_{r_c}^{\infty} r^2 u(r) dr$$

$$W_{lrc} = -\frac{2pNr}{3} \int_{r_c}^{\infty} r^2 w(r) dr$$

$$\mu_{lrc} = 4\pi\rho \int_{r_c}^{\infty} r^2 u(r) dr$$

where

$$w(r) = r \frac{du(r)}{dr}$$



## Long-Range Corrections to PBC (contd)

- Most short-range potentials decay rapidly with increasing intermolecular separations. For example, the Lennard-Jones potential asymptotes rapidly to zero for distances greater than  $2.5 \sigma$ . Therefore,  $r_c \geq 2.5 \sigma$  is an appropriate choice for the Lennard-Jones potential.
- Notice that all the long-range corrections involve both the number of particles  $N$  and the  $\rho$ . Because both these quantities are constant in the NVT-ensemble, the LRCs can be added after the simulation. However, for other ensembles in which either the number of particles or volume can change (e.g., NPT), they must be added during the simulation.



## Neighbour List

- The implementation of periodic boundary conditions reduces the computation cost significantly because calculations involving intermolecular separation greater than the cut-off distance are not included.
- In 1967, Verlet proposed that computation time could be reduced further by keeping a periodically updated list of the neighbours of each molecule. Instead of searching repeatedly for neighbouring molecules, the neighbours of each molecule are found from the list and the interactions are calculated.
- This avoids searching for molecules which do not contribute to the short-range interaction because they are beyond the cut-off separation. The algorithm below illustrates the neighbour list method.



## Neighbour List (contd)

- Neighbour List Algorithm

Part 1             $topOfList \leftarrow 0$         //start with empty list

Part 1.1        **loop**  $i \leftarrow 1 \dots N - 1$  //select molecule  $i$

$listEntry_i \leftarrow 0$

Part 2            **loop**  $j \leftarrow i + 1 \dots N$  //look for neighbours of  $i$

Part 2.1

                  Evaluate  $rx_{ij}$ ,  $ry_{ij}$  and  $rz_{ij}$

                  Evaluate periodic images (  $rx_{ij}'$ ,  $ry_{ij}'$  and  $rz_{ij}'$  )

$r^2 \leftarrow rx_{ij}'^2 + ry_{ij}'^2 + rz_{ij}'^2$

Part 2.2

**if** ( $r^2 < (rCu^2 + d)$ ) //neighbour found

$topOfList \leftarrow topOfList + 1$

$listEntry_i \leftarrow topOfList$  //position of  $j$  on list

$list_{topOfList} \leftarrow j$     //enter  $j$  on list

**end if**

**end j loop**

**end i loop**



## Neighbour List (contd)

- Starting with an empty list (Part 1), each molecule is selected in turn (Part 1.1) and a search is made for neighbours (Part 2) which involves evaluating the intermolecular separation (Part 2.1).
- If a neighbour is found it is entered on the list (Part 2.2). The creation of a neighbour list involves two arrays. The array *list* contains the neighbours of molecule *i*. The array *listEntry* stores the position of the last neighbour of molecule *i* in *list*.
- For each molecule, the *listEntry* value allows us to index the neighbouring molecules. The variable *d* is used to encompass molecules slightly outside the cut-off distance. This provides a buffer for those molecules that are likely to come within the cut-off distance in the next few time steps thereby reducing the update frequency of the neighbour list.



## Neighbour List (contd)

- The use of neighbour lists reduces the computing substantially, particularly for systems with 500 or more molecules. However, it must be updated periodically otherwise it will become inaccurate.
- An update interval of 10-20 steps is usually sufficient; the optimal number depends on the size of the cut-off distance. Alternatively, the algorithm can be extended to update the neighbour list automatically when the sum of the magnitudes of the two largest displacements exceeds the buffer distance ( $d$ ).
- The benefit of Neighbour List is often more apparent for molecular dynamics (see Module #).



## Problems

1. Write a C function (`coordinates assignCoords(int N, double L)`) that randomly assigns molecular coordinates to  $N$  atoms on the interval  $-0.5L$  to  $0.5L$ , returning them via a structure
2. Devise MC algorithm to calculate properties of hard spheres.
3. Determine the distance at which the Lennard-Jones potential has a minimum value.
4. Write a C function (`nearestInt(double rx[ij], double boxl)`) to be the equivalent of the Fortran `anint` function.
5. Determine the LRCs for energy, pressure, chemical potential for the Lennard-Jones potential.
6. Devise an algorithm that uses a neighbour list in the evaluation of the potential energy.



## 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, pages 140-180.

D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, Academic Press, San Diego, 1996, pages 19-46.

R.J. Sadus, *Molecular Simulation of Fluids: Theory, Algorithm and Object-Oriented*, Elsevier, Amsterdam, 1999, pages 131-148