3 Monte Carlo Simulations and the Photon Gas

In this set of exercises, you will be briefly introduced to the concept of importance sampling and detailed balance in Monte Carlo simulations. You shall then apply the Metropolis Monte Carlo scheme to sample a photon gas and a Lennard-Jones system to obtain their respective energies from proper phase-space averages.

3.1 Monte Carlo and the Importance of Importance Sampling

In the last exercise session, you applied the simplest Monte Carlo technique - a *random sampling* technique - to find the value of π. A generalisation of such a Monte Carlo scheme is, however, less than straightforward. Imagine you are interested in a microcanonical observable average:

$$\langle O \rangle = \frac{\int_\Gamma O(\Gamma) e^{-\beta E(\Gamma)} \mathrm{d}\Gamma}{\int_\Gamma e^{-\beta E(\Gamma)} \mathrm{d}\Gamma}. \tag{1}$$

If such an integral were to be obtained from numerical quadrature procedures, with a suitably fine mesh with M points along each degree of freedom, the calculation would become completely untractable all too quickly, since the procedure scales as $O(M^{3N})$. Furthermore, such a scheme would be associated to a large statistical error; numerical quadratures work reasonably fine for functions that are smooth on the scale of the mesh - but the Boltzmann factor is a highly oscillatory quantity. The oscillatory nature of $e^{-\beta E(\Gamma)}$ will furthermore result in many points of no importance being sampled, since their Boltzmann factor will be vanishing. The idea behind *importance sampling* lies in an extended sampling of regions where the Boltzmann factor is of considerable magnitude, with fewer sampling moves elsewhere. Very simple importance sampling schemes can be constructed, but they fail when used on multidimensional integrals, since they require analytical expressions for the partition function. If that were possible, there would be little interest in performing computer simulations - as discussed in statistical mechanics, all thermodynamic quantities can directly be determined if an analytic expression for the partition function is known.

3.2 The Concept of Detailed Balance

When calculating ensemble averages, one is often not interested in the configurational part of the partition function, but in the average instead:

$$\langle O \rangle = \frac{\int O(\mathbf{q}) e^{-\beta V(\mathbf{q})} \mathrm{d}\mathbf{q}}{\int e^{-\beta V(\mathbf{q})} \mathrm{d}\mathbf{q}}, \tag{2}$$

where we have restricted the expression to a configurational average, integrating out over all momenta. One is thus left with the *configurational* partition function, containing only the *potential* energy in the exponent, rather than the potential and kinetic term. (You can easily convince yourself that this is possible if the potential and kinetic term are not

coupled). Rewriting eq. 2 in terms of a probability density $\mathcal{N}(\mathbf{q})$ (*cf.* the Boltzmann distribution), one finds:

$$\langle O \rangle = \int O(\mathbf{q}) \mathcal{N}(\mathbf{q}) \mathrm{d}\mathbf{q} \tag{3}$$

$$\mathcal{N}(\mathbf{q}) = \frac{e^{-\beta V(\mathbf{q})}}{\int e^{-\beta V(\mathbf{q})} \mathrm{d}\mathbf{q}}. \tag{4}$$

Therefore, an ensemble average of an observable should be accessible by random sampling *if* this sampling can be carried out according to the probability distribution defined in $\mathcal{N}(\mathbf{q})$. In such a case, *on average*, the number of points generated in a volume element $\mathrm{d}\mathbf{q}$ must be equal to $L\mathcal{N}(\mathbf{q})$, where L denotes the total number of points which are generated. One may therefore rewrite the above expression in an approximate form:

$$\langle O \rangle \approx \frac{1}{L} \sum_{i=1}^{L} n_i O(\mathbf{q}_i). \tag{5}$$

The average over an observable will be given by the sum over all L points i that have been generated in the sampling, weighted each by the number of occurence n_i of state $O(\mathbf{q}_i)$. Still, one is left with the problem that the points have to be generated with a relative probability that corresponds to the Boltzmann distribution $\mathcal{N}$.

Consider a system that is in equilibrium and where all states are equally likely to occur. In order for the system to remain in its equilibrium state, every move from an old state o to a new state n must be compensated by an inverse move. If we denote the transition probability by $\mathcal{P}$, this implies:

$$\mathcal{P}(o \to n) = \mathcal{P}(n \to o). \tag{6}$$

In a system where the probability distribution is not uniform, but given by some probability distribution $\mathcal{N}$ - such as the Boltzmann distribution - instead, one will equivalently find:

$$\mathcal{N}(o)\mathcal{P}(o \to n) = \mathcal{N}(n)\mathcal{P}(n \to o). \tag{7}$$

That is, the transition matrix $\mathcal{P}$ that governs the probability of a step to occur must satisfy the above relation. Such a *detailed balance* ensures that at equilibrium, every process is equilibrated by its reverse.

3.3 The Metropolis Algorithm

There are many possible choices for $\mathcal{P}$, with maybe the most straightforward one brought forward by Metropolis *et al.* In the Metropolis Monte Carlo scheme the total transition probability $\mathcal{P}$ is expressed as a product of the probability for moving from o to n, $P'(o \to n)$, and the probability of accepting this trial move, $P_{\mathrm{acc}}(o \to n)$, such that:

$$\mathcal{P}(o \to n) = P'(o \to n) P_{\mathrm{acc}}(o \to n). \tag{8}$$

The transition matrix P' is chosen to be symmetric, such that $P'(o \to n) = P'(n \to o)$. Hence when detailed balance is maintained, the acceptance probability is:

$$\frac{P_{\text{acc}}(o \to n)}{P_{\text{acc}}(n \to o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} \tag{9}$$

$$= \frac{e^{-\beta V(n)}}{e^{-\beta V(o)}} \tag{10}$$

$$= e^{-\beta \Delta V(n \to o)}. \tag{11}$$

Many possibilities exist that account for this condition, with the choice in the Metropolis algorithm being:

$$P_{\text{acc}}(o \to n) = \begin{cases} \frac{\mathcal{N}(n)}{\mathcal{N}(o)} & \text{if } \mathcal{N}(n) < \mathcal{N}(o) \\ 1 & \text{if } \mathcal{N}(n) \geq \mathcal{N}(o), \end{cases} \tag{12}$$

where the factor 1 in the second case is due to $P_{\text{acc}}(o \to n)$ being a probability which cannot exceed a value of 1. The total transition matrix $\mathcal{P}$ is then:

$$\mathcal{P}(o \to n) = \begin{cases} \frac{\mathcal{N}(n)}{\mathcal{N}(o)} P'(o \to n) & \text{if } \mathcal{N}(n) < \mathcal{N}(o) \\ P'(o \to n) & \text{if } \mathcal{N}(n) \geq \mathcal{N}(o) \end{cases} \tag{13}$$

and

$$\mathcal{P}(n \to o) = 1 - \sum_{n \neq o} \mathcal{P}(o \to n). \tag{14}$$

The criterion whether or not to accept a trial move is inferred from the above equations and the normalisation of the probability distribution:

$$P_{\text{acc}}(o \to n) = e^{-\beta[V(n) - V(o)]} \tag{15}$$

$$< 1. \tag{16}$$

If $V(n) < V(o)$, the move is always accepted. However, after a move for which $V(n) > V(o)$, a random number is generated out of a uniform distribution in the interval $[0, 1]$, such that the interval spans the same range as the Boltzmann factor. The probability that the random number X is less than $P_{\text{acc}}(o \to n)$ is equal to $P_{\text{acc}}(o \to n)$ itself:

$$P\left(X < P_{\text{acc}}(o \to n)\right) = P_{\text{acc}}(o \to n). \tag{17}$$

The trial move is then only accepted if $X < P_{\text{acc}}(o \to n)$, and rejected otherwise. This scheme guarantees that the probability of accepting some trial move $o \to n$ is equal to the probability $P_{\text{acc}}(o \to n)$. Thus, the system moves towards an equilibrium distribution ($P_{\text{acc}} = 1$ for new states lower in energy). Once equilibrium is reached, it is ensured to retain the equilibrium distribution (P_{acc} according to the Boltzmann factor). The overall acceptance probability is:

$$P_{\text{acc}}(o \to n) = \min\left(1, \frac{P_{\text{acc}}(o \to n)}{P_{\text{acc}}(n \to o)}\right) \tag{18}$$

$$= \min\left(1, e^{-\beta[V(n) - V(o)]}\right), \tag{19}$$

i.e. the acceptance probability is 1 if the Boltzmann factor exceeds 1, and it is the Boltzmann factor itself otherwise. This guarantees that the sampling preserves the equilibrium distribution, *i.e.* it fulfills *detailed balance*.

3.3.1 An example of a Metropolis Monte Carlo Algorithm

a) Select a random starting configuration with coordinates $\mathbf{q}$.

b) Randomly displace the configuration, such that $\mathbf{q}' = \mathbf{q} + \Delta$

c) Accept this move with a probability of:

$$P_{\text{acc}}(o \to n) = \min\left(1, e^{-\beta[V(\mathbf{q}')-V(\mathbf{q})]}\right), \tag{20}$$

depending on whether $e^{-\beta[V(\mathbf{q}')-V(\mathbf{q})]}$ is greater or lower than 1. Keep the old configuration if the move is rejected.

d) Update the averages, regardless whether the step was accepted or not.

e) Go to b)

3.3.2 Ensemble Averages from the Metropolis Monte Carlo Algorithm

3.3.3 The Photon Gas

In this exercise you will be applying the Metropolis Monte Carlo algorithm to calculate the state occupancy of a photon gas. The photon gas is a gas-like collection of photons, which has many of the same properties of a conventional gas such as pressure, temperature and entropy. The most common example of a photon gas in equilibrium is black-body radiation. Black-body radiation is an electromagnetic field constructed by a superposition of plane waves of different frequencies, with the caveat that a mode may only be excited in units of $\hbar w$. This fact leads to the concept of photons as quanta of the electromagnetic field, with the state of the field being specified by the occupancy $\langle n_j \rangle$ of each of the modes or, in other words, by enumerating the number of photons with each frequency.

The ensemble average of the state occupancy $\langle n_j \rangle$ of a photon gas can be calculated analytically. Deriving the total energy of an idealised photon gas from quantum mechanics we know that U can be written as the sum of the harmonic oscillation energies:

$$U = \sum_{j=1}^{N} n_j w_j \hbar = \sum_{j=1}^{N} n_j \epsilon_j, \tag{21}$$

where ϵ_j is the energy of state j, n_j is the occupancy of state j ($n_j \in 0, 1, 2, \cdots, \infty$), N is the total number of photons and w is the oscilator frequency.

In this exercise, you are going to compute the ensemble average of the occupancy $\langle n_j \rangle$. The scheme you will employ is as follows:

a) Start with an arbitrary n_j.

b) Decide to perform a trial move to randomly increase or decrease n_j by 1.

c) Accept the trial move with probability:

$$P_{acc}(o \to n) = \min\left(1, e^{\beta(U(n)-U(o)}\right), \tag{22}$$

where $U(n)$ and $U(o)$ are the energies of the new and old states respectively.

d) Update averages regardless of acceptance or rejection.

e) Go to step b).

3.3.4 Coding the C++ Program

Begin by creating a photon.cpp file and enter the following code:

```
1  #include <iostream>
2  #include <ctime>
3  #include <cstdlib>
4  #include <random>
5  #include <math.h>
6
7  using namespace std;
8
9  mt19937 mt_rand;
10
11 double random(double lower, double upper) {
12     uniform_real_distribution<double> dist(lower, upper);
13     return dist(mt_rand);
14 }
15
16 int main(int argc, char * argv []) {
17
18   int numberOfIterations = atoi(argv[1]);
19   double beta = atof(argv[2]);
20   int seed = time(nullptr);
21   mt_rand.seed(globalSeed);
22
23   int trialnj = 1;
24   int currentnj = 1;
25   int njsum = 0;
26   int numStatesVisited = 0;
27
28   /***** MODIFICATION *****
```

```
29
30    Metropolis algorithm implementation to calculate <n_j>
31    Tasks:
32    1) Loop from int i = 0 to numberOfiterations
33    2) Call random(0, 1) to perform a trial move to randomly increase
34       or decrease trialnj by 1.
35       Hint: use trialnj = currentnj + 1;
36    3) Test if trialnj < 0, if it is, force it to be 0
37    4) Accept the trial move with probability defined in section 3.4.3
38       Note: Accepting the trial move means updating current sample (currentnj)
39       with the new move (trialnj);
40    5) sum currentnj and increase numStatesVisited by 1
41
42    *** END MODIFICATION ***/
43
44    double estimatedOccupancy = (double) njsum/numStatesVisited;
45
46    cout << "Average Occupancy: " << estimatedOccupancy << endl;
47    cout << "Theoretical Value: " << ( 1 / (exp(beta) - 1)) << endl;
48    cout << "Relative Error: " << fabs((exp(beta) - 1.0) * (( sum / count ) -
49      (1.0 / (exp(beta) - 1.0))));
50    return 0;
51  }
52  }
```

Here, your task is to implement the Metropolis Monte Carlo algorithm to estimate the ensemble average of state occupancy, $\langle n_j \rangle$. The code is commented thoroughly to provide you with hints. You can call the function `exp(x)` to calculate exponentials and the function `random(min, max)` to create a uniformly distributed rational number between `min` and `max`. Finally, the operators `++` and `--` provide a shorthand for incrementing and decrementing a variable by 1 as follows: `numStatesVisited++;`.

3.3.5 Compiling Your C++ Code

To compile the photon gas program, navigate your terminal focus to the directory which contains `photon.cpp` and type the following:

```
g++-4 -std=c++11 photon.cpp -o photon.x
```

This will produce an executable file called `photon.x` in your current directory. You can execute this program using the following command-line arguments:

```
./photon.x numberOfIterations betaValue
```

3.4 Configurational Sampling using the Metropolis Monte Carlo Algorithm

3.4.1 The Lennard-Jones Potential

In this exercise you will study the configuration of a collection of gaseous particles using the Metropolis Monte Carlo algorithm. The system includes N particles within a

cubic box of volume V at a given temperature T, in any configuration permitted by the Lennard-Jones potential:

$$U(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] & \text{if r} \leq \text{r}_c \\ 0 & \text{if r} > \text{r}_c. \end{cases} \tag{23}$$

This potential traditionally has an infinite range, however, the potential decays rapidly with separation distance and can be effectively ignored at large $|r|$, resulting in a faster calculation. In practical applications it is customary to establish a cutoff r_c and disregard pairwise interactions separated beyond this radius. This truncation leads to a discontinuity in the pairwise potential energy function; large numbers of these events are likely to spoil energy conservation thus an improvement is to shift the potential such that the energy continuously approaches zero at r_c:

$$U(r) = \begin{cases} 4\epsilon \left(\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] - \left[\left(\frac{\sigma}{r_c}\right)^{12} - \left(\frac{\sigma}{r_c}\right)^{6} \right] \right) & \text{if } r \leq r_c \\ 0 & \text{if } r > r_c. \end{cases} \tag{24}$$

This approach results in a potential that produces discontinuities in the first and higher order derivatives. To compensate, switching functions are often employed to smoothly and continuously taper the pair potential to zero between two cutoff limits.

Truncating pair interactions systematically removes a non-trivial contribution to the net potential energy and pressure. For interactions that are cut but not shifted, one can approximately add the interactions beyond r_c to the total energy and pressure, assuming the radial distribution function $g(r > r_c) \approx 1$:

$$U = U_{pairs} + U_{tail} \tag{25}$$

$$P = P_{pairs} + P_{tail}, \tag{26}$$

where

$$U_{tail} = \frac{8\pi N^2}{3V}\epsilon\sigma^3 \left[\frac{1}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3 \right] \tag{27}$$

$$P_{tail} = \frac{16\pi N^2}{3V}\epsilon\sigma^3 \left[\frac{2}{3}\left(\frac{\sigma}{r_c}\right)^9 - \left(\frac{\sigma}{r_c}\right)^3 \right]. \tag{28}$$

To sample configurational space using the Lennard-Jones potential, a randomly selected particle is first randomly translated to generate a new system configuration. Whether the new configuration is accepted depends on the acceptance probability discussed in section 3.3.1. This procedure repeats iteratively such that classical phase space is directly sampled and ensemble averages of physical properties become arithmetic averages over their sampled values.

3.4.2 Using and Compiling the LJ-NVT Code

In this exercise the code is provided for you. To compile the code navigate your terminal to the `Source` directory and type `make`. This will produce an executeable file called `mc_nvt`. To run the program, navigate your terminal to the `Run` directory, and type `./run`. The program will print results to your terminal, and also produce a file called `movie.pdb` in your `Run` directory.

3.5 Exercises

3.5.1 Monte Carlo and Statistical Mechanics

a) Show that based on the form of the Hamiltonian:

$$\mathrm{H} = \mathrm{T}(\mathbf{p}) + \mathrm{V}(\mathbf{q}), \tag{29}$$

the partition function can be divided into a kinetic and potential part, and for an ensemble average of an observable $O(\mathbf{q})$ that depends on $\mathbf{q}$ only, one has:

$$\langle O(\mathbf{q}) \rangle = \frac{\int_\Gamma O(\Gamma) e^{-\beta E(\Gamma)} \mathrm{d}\Gamma}{\int_\Gamma e^{-\beta E(\Gamma)} \mathrm{d}\Gamma} \tag{30}$$

$$= \frac{\int O(\mathbf{q}) e^{-\beta V(\mathbf{q})} \mathrm{d}\mathbf{q}}{\int e^{-\beta V(\mathbf{q})} \mathrm{d}\mathbf{q}}, \tag{31}$$

which is the equation you have encountred in the section on detailed balance.

b) **Bonus:** In the Metropolis scheme, why is it important that P' be a symmetric matrix?

3.5.2 The Photon Gas

a) How can this scheme retain detailed balance when $n_j = 0$? Note that n_j cannot be negative.

b) Make modifications in the code, within the section '`MODIFICATION` $\cdots$ `END MODIFICATION`'. Include your entire code within your report and comment upon the part that you wrote.

c) Using your code, plot the photon-distribution (average occupation number as a function of $\beta\epsilon \in [0.1, 2]$). Assume that the initial $n_j = 1$ and $\epsilon_j = \epsilon$ and recalculate with the same $\beta\epsilon$ values using the analytical solution:

$$\langle N \rangle = \frac{1}{e^{\beta\epsilon} - 1}. \tag{32}$$

Plot your calculated values versus those from the analytical solution and include your curve in your report.

d) **Bonus:** Modify the program in such a way that the averages are updated only after an accepted trial move. Why does ignoring rejected moves lead to erroneous results? Starting from $P_{acc}(o \to n)$, define $P'(o \to o)$ (*i.e* the probability that you stay in the old configuration). Recall that the transition probability P' is normalised.

e) **Bonus:** At which values of β does the error you obtain when ignoring rejected moves become more pronounced and why?

3.5.3 Sampling Configurational Space

a) Perform a simulation at $T = 2.0$ and various densities $\rho \in [0.05, 1.0]$. Up to which density does the ideal gas law hold?

$$\beta P = \rho \tag{33}$$

Note: you can modify T and ρ in the `input` file within the `Run` directory. The value of $\langle P \rangle$ is printed at the end of the calculation.

b) The program produces a sequence of snapshots of the state of the system. Visualise these snapshots using the program Visual Molecular Dynamics (VMD). See section 3.5.4 for instructions on using VMD. Include snapshots in your report at the following iteration numbers: 0, 30, 50, 500, 999. Explain the observed behaviour of the system upon increasing iteration number.

c) Instead of performing a trial move in which only one particle is displaced, one can do a trial move in which all particles are displaced. What do you expect will happen to the maximum displacements of these moves when 50% of all displacements are accepted?

3.5.4 Visual Molecular Dynamics (VMD)

To open a trajectory or structure in VMD, navigate your terminal to the file of interest and type the following command:

```
/Applications/VMD/VMD\ 1.9.1.app/Contents/vmd/vmd_MACOSXX86 movie.pdb
```

You can explore the system by holding your left mouse button and moving. You can translate the system by pressing `T` or selecting `Mouse` → `Translate Mode` in the VMD Main panel. Rotation and scaling can be performed from this menu as well, or you can just type `r` and `s` respectively. To change the centre of the viewport (*i.e* the position about which geometry operations occur), press `c` and select a particle to move the centre to.

To view the trajectory as a movie, press the play button from the VMD Main panel, or manually move the counter bar to scroll through the trajectory. Experiment with the buttons on this panel to become more comfortable - you will need this program for the following exercises.

Changing the Drawing Mode Click on `Graphics` in the VMD Main panel, and go to `Representations....` On the `Draw style` tab, select `ColourID` from the `Colouring Method` drop-down box, and select a colour of your choosing. Next, select `CPK` from the `Drawing Method` drop-down box, and change `Sphere Scale` to 0.1 and `Sphere Resolution` to 20. Click `Apply` and close the `Graphical Representations` window.

Changing the Viewport Background Click on the `Graphics` item in the VMD Main panel, and select the `Colours...` option. Next, click on the `Display` item within the `Categories` list, and select `Background`. Change the background of the viewport to `8 white`. Close the `Colour Controls` window.

Rendering and Saving To save a snapshot of the current frame, go to `File` in the VMD Main panel, and click the `Render...` option. Change the filename to something meaningful, and click `Start Rendering`. The mac program `Preview` will immediately open, and from there you can export the snapshot as a `.png` image.