

Lecture 2:

Direct Calculation of the Average Energy in the Classical Limit

In the canonical ensemble (a system with a constant number of molecules N , confined in a volume V , at temperature T), the expectation value of a quantum-mechanical operator \hat{X} , corresponding to the observable X , is at equilibrium given by:

$$\langle X \rangle = \sum_i X_i p_i = \sum_i X_i g_i e^{-E_i/kT} / Q_N, \quad (4)$$

where $X_i = \int_{\text{All space}} \phi_i * \hat{X} \phi_i$ is the expectation value of \hat{X} in the (normalised) state ϕ_i . The states ϕ_i

are solutions to the time-independent Schrödinger equation $\hat{H}\phi_i = E_i\phi_i$. Equation (4) suggests how we should go about computing thermal averages: first we solve the Schrödinger equation for the (many-body) system of interest, and next we compute the expectation value of operator \hat{X} for all those quantum states that have a non-negligible statistical weight. Unfortunately, this approach is doomed for all but the simplest systems. Fortunately, this equation can be simplified to a more workable expression in the classical limit, in which the state of N particles is described by specifying the instantaneous positions and momenta of each particle. This is a $6N$ -dimensional space $\Gamma : (r_1, \dots, r_N; p_1, \dots, p_N) \equiv (r^N, p^N)$ usually called phase space. Classically, the sum over states i is a integration performed over phase space

$$\sum_i \rightarrow \frac{1}{N!h^{3N}} \int_{\Gamma} dr^N dp^N,$$

where h is Planck's constant. The factor h^{3N} considers the “graininess” of the phase space, and $N!$ considers the over-counting of states due to permutation of identical particles. The classical expression obtained from equation (4) for the average of a quantity X in a system of N atoms is then given by:

$$\langle X \rangle = \frac{\int dp^N dr^N \exp\left\{-1/(kT)\left[\sum_i p_i^2/(2m_i) + V(r^N)\right]\right\} X(p^N, r^N)}{\int dp^N dr^N \exp\left\{-1/(kT)\left[\sum_i p_i^2/(2m_i) + V(r^N)\right]\right\}} \quad (5)$$

where $V(r^N)$ is the potential energy of the particles. Furthermore, the classical expression for the canonical partition function obtained from equation (2) is given by:

$$Q_N = \frac{1}{N!h^{3N}} \int dp^N dr^N \exp\left\{-1/(kT)\left[\sum_i p_i^2/(2m_i) + V(r^N)\right]\right\} \quad (6)$$

Example To illustrate using equation (5), consider calculating the average kinetic energy of a molecule of mass m moving in one dimension, but doing so in contact with a heat bath at temperature T so that it undergoes random thermal motion. If p is the momentum at any instant, the average kinetic energy $\langle E \rangle_k$ is found with $X(p) = p^2 / 2m$, and the ratio of integrals is,

$$\langle E \rangle_k = \frac{1}{2m} \frac{\int_{-\infty}^{\infty} p^2 e^{-p^2/(2mkT)} dp}{\int_{-\infty}^{\infty} e^{-p^2/(2mkT)} dp} = \frac{1}{2} kT. \quad (7)$$

Consider next either the classical harmonic oscillator or the quantum harmonic oscillator at high temperature. There is a kinetic energy contribution to the total energy because the atoms are moving relative to one another about their equilibrium position with velocity v . Equation (7) gives the average kinetic energy. Additionally, there is the potential energy $X(x) = V = k_f x^2 / 2$, where x is the displacement from the equilibrium position and k_f is a force constant. Mathematically this is the same calculation as equation (7) and produces the same result, but x is used instead of p . The total average energy is therefore kT . This result is a consequence of the Equipartition Theorem of classical mechanics where each quadratic term in the energy representing either the momentum p (kinetic energy $E = p^2 / 2m$) or position x , contributes $kT/2$ to the total energy.

The Metropolis Algorithm

The Metropolis algorithm is a general Monte-Carlo method that allows various average properties to be calculated, such as $\langle E \rangle$ or $\langle E^2 \rangle$, by randomly sampling configurations and retaining mainly those that are important contributors to the average value. This method of sampling only important contributions is (naturally) called importance sampling. In the Metropolis Algorithm because the Boltzmann distribution is usually used, the sampling is energy biased.

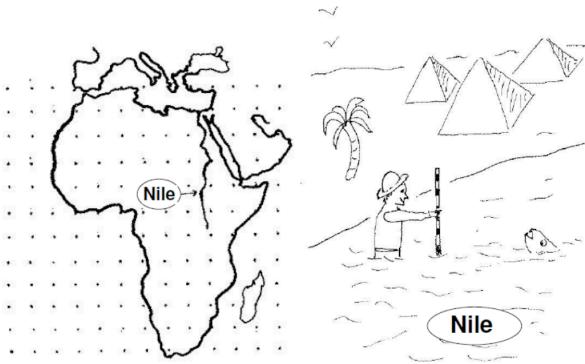


Figure 3. Measuring the depth of the Nile: a comparison of conventional quadrature (left) with the Metropolis scheme (right).

To obtain the average $\langle X \rangle$, as defined in equation (5), either the two integrals can be determined separately, or their ratio determined. The unique feature of the Metropolis algorithm is that it allows an estimation of the *ratio* of the integrals to be made in an efficient way, and hence, averages are determined directly. This avoids having to calculate Q_N which, as we have seen, may be impossible. The algorithm overcomes the necessity to search all configurations by using the Boltzmann distribution of energies to bias the random choice of which configuration to add to the total. In doing this, the algorithm tries to add to the estimate of $\langle X \rangle$ only those configurations contributing significantly to its value. In effect, the algorithm performs a random walk among the configurations, i.e. by changing the velocity or position. This random walk is also called a Markov process, which is defined as a process whereby a ‘system’ goes from one state to another in a random fashion but has no memory of its previous condition.

To sample points according to the Boltzmann distribution, it is sufficient, but not necessary, to impose ‘detailed balance’ between any two configurations. This means that if w_{12} is the rate of going from configuration (or state) 1 with energy E_1 , to a new one 2 with energy E_2 , then the reverse transition w_{21} is related to it by

$$\frac{w_{12}}{w_{21}} = \frac{e^{-E_2/kT}}{e^{-E_1/kT}} = e^{-\Delta E/kT}, \quad \Delta E = E_2 - E_1. \quad (8)$$

The ratio of rates is thus the same as the ratio of probabilities of moving between two configurations. Detailed balance, equation (8), ensures that any one configuration can be reached from any other in a finite number of steps. This behaviour is called ergodicity and it means that no configurations are systematically missed. (Alternatively, this can be expressed by saying that the time average is the same as the ensemble average).

The essential step in the Metropolis algorithm is the method of choosing the ratio of probabilities. Two energies E_1 and E_2 from two guessed configurations, the current one 1, and a new one, 2, are calculated. If the new one has the lower energy then this is accepted. If not, the chance of 2 contributing to $\langle X \rangle$ is guessed by comparing $e^{-(E_2-E_1)/kT}$ to a uniform random number in the range 0 to 1, and accepting 2 if the random number is smaller. If not, the current configuration 1 is used again.

Outline of the Metropolis Algorithm

The algorithm to calculate an average quantity $\langle X \rangle$ is summarised as:

- (1) Initialise parameters, e.g. $X_{tot} = 0$

Calculate E_1 using any reasonable initial values.

Calculate the initial X_1 .

- (2) Start a loop

Calculate E_2 using other parameter values chosen at random.

(i) If $E_2 < E_1$, keep the new state 2 and add X_2 to a total quantity X_{tot} .

(ii) If $E_2 > E_1$, then calculate $e^{-(E_2-E_1)/kT}$:

(a) If $e^{-(E_2-E_1)/kT} > r$, where r is a uniform random number between 0 and 1, keep the new state 2 and add X_2 to X_{tot} .

(b) If $e^{-(E_2-E_1)/kT} \leq r$, retain the old state 1 and add X_1 to X_{tot} .

(3) Continue the loop until X_{tot} is obtained to a sufficient accuracy or a predetermined number of trials are completed.

(4) Average $\langle X \rangle = X_{tot} / N$ for N calculations.

Notice that even if a new state is not accepted, the last used value X_1 is still added to the total. This is because each (randomly chosen) configuration has to be added in. If a sufficient number of samples are used in the calculation, all configurations will be sampled in proportion to their importance in contributing to $\langle X \rangle$. As an illustration, the mean displacement, $\langle x \rangle$, the mean square displacement $\langle x^2 \rangle$, the mean potential energy and the vibrational heat capacity of a classical harmonic oscillator are calculated in the lecture.

Fluctuations

To calculate $\langle E \rangle$, a Monte Carlo method can also be used. Just as in an experiment where repeated measurements improve the precision, in a Monte Carlo calculation increasing the number of trials acts similarly. To illustrate the difference between a single vs. many measurements, the standard deviation σ of the average energy $\langle E \rangle$ can be used. This is defined as

$$\sigma = \sqrt{\langle E^2 \rangle - \langle E \rangle^2}, \quad (9)$$

and the fractional error is

$$\frac{\sigma}{\langle E \rangle} = \frac{\sqrt{kT^2 C_V}}{\langle E \rangle}. \quad (10)$$

In order to obtain Equation (10) we used the fact that the heat capacity $C_V = \left(\frac{\partial U}{\partial T} \right)_V$ of a system, in combination with Equation (3), can be expressed as

$$C_V = \frac{1}{kT^2} \left(\langle E^2 \rangle - \langle E \rangle^2 \right). \quad (11)$$

Furthermore, we used that $\langle E^2 \rangle_k = 3(kT/2)^2$, which can be calculated similarly to the average energy illustrated above (see Eq. (7)). For a single particle $C_V = k/2$ and $\langle E \rangle = kT/2$ (this can be calculated but is also known from the equipartition theorem), making $\sigma/\langle E \rangle = 1$. Thus the average energy obtained from a measurement on a single particle has, not surprisingly, a very large fractional error. When N particles are measured (or N repeated measurements of the same one) then $C_V = Nk/2$, $\langle E \rangle = NkT/2$, and now $\frac{\sigma}{\langle E \rangle} \approx \frac{1}{\sqrt{N}}$, which is a vanishingly small quantity if N is large, and a good estimate of the true $\langle E \rangle$ is obtained. Similarly, with a Monte Carlo simulation, the more samples that are included, the more accurate the result becomes and the standard deviation improves in proportion to $1/\sqrt{N}$.

Problem 2: Workshop on Metropolis Algorithm of the Harmonic Oscillator

The harmonic oscillator with force constant k_f has potential energy $V = k_f x^2/2$ at displacement x from its minimum energy. To make the calculation classical, many energy levels must be populated at temperature T . This implies that a very high temperature or a very small force constant should be used such that k_f/T is small, for example 0.05. A temperature of 300 K and $k_f = 10 \text{ N m}^{-1}$ is used. This force constant is far smaller than that for many molecules although alkali metal dimers do have small force constants.

To calculate the average energy, random guesses of the values of the displacement are made. To allow complete sampling of the distribution, the maximum displacement must be large enough to make $e^{-V/kT}$ small, $\approx 10^{-8}$ or less. In addition, the range of x must be positive as well as negative, and the random number generator must ensure this. If this is not done, then microscopic reversibility is not achieved because not all the possible configurations, in this case bond extension, can be sampled.

Pseud code for the Metropolis algorithm: The Harmonic Oscillator

The code is written in a basic form so that it may more easily be translated into other languages such as Python.

Notes:

- (1) The instruction `rand()` generates a random number between 0 and 1.

```

N = 20000                                # number of samples

deltax = 0.15                             # max displacement nm

nm = 10^(-9)                               # nanometres

kf = 1.381*10^(-23)                        # Boltzmann const J m^-2 K^-1

T = 300.0                                  # temperature K

kT = k*T                                    # initial thermal energy

kf = 10.0                                   # force const N m^-1

V(x)-> kf*x^2*nm^2/2                      # def function pot'l energy kg m^2 s^-2

Etot = 0.0                                  # initial energy <E>

E2tot = 0.0                                 # initial <E^2>

x1 = 0.0                                    # first guess of x in nm

E1 = V(x1)                                 # first guess of energy

for i from 1 to N do                      # start loop step (2)

    x2 = x1 + rand()*deltax                # new x position

    E2 = V(x2)                             # new PE

    DeltaE = E2 - E1                      # energy difference

# next line is Metropolis sampling

    if DeltaE <= 0.0 then

        x1 = x2                            # save new configuration

        E1 = E2                            # save new energy

    else if DeltaE > 0.0 then

        if exp(-DeltaE/kT) > rand() then

            x1 = x2                            # save new configuration
            E1 = E2                            # save new energy

    end if

```

```

end if

Etot = E1                                # always add to total <E>

E2tot = E2tot + E1^2                      # add to total <E^2>

end do                                     # end loop step (3)

# average step

Eav = (Etot)                               # <E>

E2av = (E2tot/N)                          # <E^2>

CV = (E2av-Eav^2) / (k*T^2)

```

Please note that there are (about four minor) mistakes in the code that you need to find!

Average potential energy and heat capacity

A typical calculation produces 2.05×10^{-21} J for the average potential energy (E_{av} , see Maple code), 1.27×10^{-41} J² for its square (E_{2av}), and 6.88×10^{-24} J K⁻¹ for the heat capacity (CV). These will vary slightly each time the calculation is run, but when the number of trials is large, ≈ 20000 , they are always close to the theoretical values (per molecule) which are $\langle E \rangle = 2.07 \times 10^{-21}$ J, $\langle E^2 \rangle = 1.28 \times 10^{-41}$ J², and $C_V = 6.90 \times 10^{-24}$ J K⁻¹ at 300 K.

Note that in this implementation and only for simplicity, adding E_1 , E_1^2 , x_1 or x_1^2 to their respective totals starts immediately without allowing the random walk to come close to its equilibrium value, which takes several hundred steps. This introduces an error, but as the number of Monte-Carlo steps is large at 20000, the error is small.

Task 1: Average energy and heat capacity (4 marks)

- a) The maple code is essentially structured in four parts: 1. Initialisation, 2. Monte Carlo loop, 3. Data calculation, and 4. analysis. Indicate these four parts in the example code above. (0.5 mark)
- b) State what are the mistakes in the Monte Carlo code above? (0.5 mark)
- c) Use the Monte Carlo code to calculate the average energy, the average square energy, and the heat capacity of the harmonic oscillator. (2 mark)
- d) Verify that the theoretical value of the average potential energy is $\langle E \rangle = 2.07 \times 10^{-21}$ J. To do this you need to use Eq. (5) and perform both integrals using the numerical integration routine in Python (1 mark)

Task 2: Average displacement

The average displacement can be calculated by adding

lines (**1 marks**)

```
Xtot = Xtot + x1          # add to total <X>  
X2tot = X2tot + x1^2      # add to total <X2>
```

in the loop after `E2tot:=...` and remembering to define and set the new variables to zero at the start of the calculation.

a) What is the typical value for $\langle x \rangle \equiv X_{tot} / N$? Why is this result expected? (0.5 mark)

b) What is the typical value for $\langle x^2 \rangle \equiv X_{2tot} / N$? How does the result compare to the theoretical value of $kT / k_f = 0.00041 \text{ nm}^2$? (0.5 mark)

Task 3: Understanding (1 marks) Explain why Metropolis Monte Carlo method works, in particular why and how only important contributions to the statistical average are obtained in the simulations?

Task 4: Error calculations (1 mark) Calculate the fractional error for the average energy per oscillator using eq. (10) for several values of the number of measurements N (i.e. 10,100,1000,...). Generate a corresponding plot.

Task 5: Quality of presentation of report in Jupyter notebook (2 marks)

Task 6: Bonus point (1 mark). Justify why your work/report goes beyond just completing the tasks.

Submission of work:

- The work must be submitted as an ipynb on minerva
- The filename must be Surname-problem2.ipynb
- The self-assessment must be added at the end of the ipynb

Self-assessment:

- Aim is that you give an honest reflection on how you completed the tasks
- You will need justify all your marks
- Declaration of integrity (That the work you submitted is yours, and that you marked it fairly)

Deadline: 16th Feb. 2026, 5pm