Project 1: Temperature of an Einstein solid

Dr. Vladimir Skokov
Phys 6240

1 Temperature and an ideal thermometer

The temperature of a system is an abstract quantity in part because we don't have a direct way to measure it. Instead, we need to measure some other quantity as the pressure in a container or the expansion of a metal, and then calibrate these measurements with the temperature by assuming that the measurements change in some known way with temperature. We now discuss a simple thermometer which gives a direct measure of the temperature. This thermometer has only one property, its energy, and can be realized in a computer simulation. An ideal thermometer interacts very weakly with the system of interest, but strongly enough to obtain an accurate measurement. Also there needs to be a known relation between some property of the thermometer (in our case energy) and the temperature.

1.1 Computer simulations: "demon" algorithm

Consider a system of one particle which we will call a *demon* that can exchange energy with another system. The demon will play a role of a thermometer, and the other system, which we will call "the system", is much larger and plays a role of a heat bath. We will use the *demon* to measure the temperature of the heat bath. As long as almost all of the energy resides in the heat bath, the temperature of the heat bath will not change. The demon obeys the following rules or algorithm:

1. Set up an initial microstate of the system with the desired total energy E and assign an initial energy E_d to the demon. For convenience the initial demon energy is usually set to zero, $E_d = 0$.

2. Make a trial change in the microstate. For example, for the Einstein solid (see below), choose a particle at random and randomly increase or decrease its energy by one. For the Ising model, flip a spin chosen at random.

3. Compute the change in energy of the system, ΔE. If $\Delta E \leq 0$, accept the change, and increase the energy of the demon by $|\Delta E|$. If $\Delta E > 0$, accept the change if the demon has enough energy to give to the system (i.e. after subtraction $E_d \geq 0$), and reduce the demon's energy by ΔE. The only constraint is that the demon's energy must remain greater than a lower bound which we take to be zero. If a

trial change is *not* accepted, the existing microstate is counted in the averages. In either case the total energy E of the system plus the energy E_d of the daemon remains constant.

4. Repeat steps 2 and 3 many times (Run many trials.).

5. Compute the averages of the quantities of interest once the system and the daemon have reached equilibrium.

2 An Einstein solid

Consider a system of N noninteracting distinguishable particles such that the energy of each particle is restricted to integer values, that is, $\epsilon_n = 0, 1, 2, 3, \ldots$. For the reasons not discussed here, we will refer to this model as an Einstein solid (a cruder version of the Debye model, see the textbook).

2.1 The demon and the Einstein solid

Consider a system that exchanges energy with an Einstein solid of N particles. The demon selects a particle at random and randomly changes it energy by ± 1 consistent wit the constraint that $E_d \geq 0$. In this case the energy of of the particle in the system also must remain non-negative! If a trial change is not accepted, the existing microstate is counted in all averages. Write a computer program to perform numerical simulations.

Thus the algorithm (repeated)

1. Set up an initial microstate with the desired total energy and assign an initial energy to the demon. For simplicity, we will choose the initial demon energy to be zero and assign the total desired energy (which must be an integer) to one of the particles.

2. Choose a particle at random and increase or decrease its energy by ± 1. If $\Delta E = -1$, we accept the change, and give energy $+1$ to the demon so that the total energy of the system plus the demon is constant. If $\Delta E = 1$, we accept the change if the demon has enough energy to give to the system, and reduce the demon's energy by 1. If the trial change is not accepted, the existing microstate is counted in the averages.

3. Repeat step 2 many times.

4. Compute the averages of the quantities of interest once the system and the demon have reached equilibrium.

Problems:

1. (N) Choose $N = 50$ and $E = 200$. What is the mean energy of the demon after the equilibrium between the daemon and the system has been established? Compare

the values $\bar{E}_d$ and $\bar{E}/N$. Increase/decrease E. Plot dependence of $\bar{E}_d$ on $\bar{E}/N$ for large $E/N \gg 1$.

2. (N) Consider small values of E/N (keep $N = 50$), plot $\bar{E}_d$ as a function of $\bar{E}/N$ and determine if there is a simple relation between $\bar{E}_d$ on $\bar{E}/N$.

3. (N) Compute the probability density $p(E_d)$ for various values of E and N. Fit your results to the form $p(E_d) \propto \exp(-\beta E_d)$, where β is a parameter. Does the form of $p(E_d)$ depends on E and N? Plot your results!

4. (A) Explain why the form of $p(E_d)$ is given by the Boltzmann distribution. What property of the demon appears to be universal?

5. (A) Analytically derive the demon's mean energy as a function of temperature.

6. (A) Invert the above to find $T(\bar{E}_d)$.

7. (N) Do numerical simulations for different E and fixed $N = 50$. Determine $\bar{E}_d$. Plot T as a function of E.

8. (N) Do numerical simulations for different N and fixed $E = 200$. Determine $\bar{E}_d$. Plot T as a function of N.

(N) $\equiv$ perform numerical simulations and show the plots.
(A) $\equiv$ perform analytical computations.

3 Example of a Python program

First, include all needed libraries

```
from numpy import *
import random as Rand
```

Next, define the number of particles and the total energy

```
Initial_energy=200
Number_of_particles=50
```

"E" will be an array of particle energies and "Edemon" is the energy of the demon.

```
E=zeros(Number_of_particles);
Edemon=200;
```

Here we have a function that randomly chooses a particle, N, and DeltaE=$\pm$1. We check if the energies of particle and demon are positive after we add and subtract DeltaE respectively. If they are we accept the change and return 0. Otherwise we do not accept change and return 1.

```
def single_demon_sweep():
    global Edemon
    N = Rand.randint(0,Number_of_particles-1)
    DeltaE = 2*Rand.randint(0,1)-1
```

```
    E_plus_DeltaE = E[N]+DeltaE
    Edemon_minus_DeltaE = Edemon-DeltaE
    if E_plus_DeltaE*Edemon_minus_DeltaE>=0:
            E[N] = E_plus_DeltaE
            Edemon = Edemon_minus_DeltaE
            return 0
    return 1
```

We call the function many time to thermalize the system

```
  for i in range(10000):
    single_demon_sweep()
```

Now we can calculated observables

```
E_average=0;
N_trials=0;
for i in range(100000):
    trial = single_demon_sweep()
    E_average=E_average+sum(E)
    N_trials=N_trials+1
    print Edemon
print E_average/N_trials
```