13 Metropolis MC simulation

(this section is almost identical to part 9 of my MC simulation course).

We have seen on this course that MD can be used to simulate all of the following thermodynamic equilibrium ensembles:

NVE	microcanonical	isolated
NVT	canonical	system in temperature bath
NPT	isothermal-isobaric	constant T and pressure P
μVT	grand canonical	constant chemical potential and T

There is also an entirely different formalism which can be used to sample these: Monte Carlo simulations based on the Metropolis algorithm.

Let us recall the basic difference of the two approaches from the theory section. In MD the desired quantity is obtained from a time average of a quantity, $\langle \mathcal{A} \rangle_t$. In MC simulations the desired thermodynamic quantities are determined as ensemble averages, as in statistical mechanics. A desired quantity $\mathcal{A}$ is obtained as

$$\langle \mathcal{A} \rangle_{\text{ens}} = \frac{\displaystyle\sum_{\mathbf{\Gamma}} A(\mathbf{\Gamma}) \rho_{\text{ens}}(\mathbf{\Gamma})}{\displaystyle\sum_{\mathbf{\Gamma}} \rho_{\text{ens}}(\mathbf{\Gamma})} \tag{1}$$

So the idea is to obtain an average over points in phase space $\mathbf{\Gamma}$ which are "frozen in time". We do not need to simulate any real time dependence of the system, only construct a sequence of states in phase space in the correct ensemble. It is important to keep in mind that the value of the quantity $\mathcal{A}$ will depend on which ensemble is dealt with.

The basic idea of Metropolis MC is to obtain thermodynamic equilibrium averages. It is entirely unsuitable to simulate nonequilibrium systems.

13.0.1 Importance sampling

To get a thermodynamic ensemble average one needs to calculate eq. 1. But often most points have very little weight. This is clear if you for instance remember that the weight function for the NVT ensemble is the Boltzmann distribution

$$\rho_{NVT} \propto e^{-E/k_B T}$$

where E is the total energy. In principle any value of E is possible, but the probability to reach a state with say $E = 1000 k_B T$ is vanishingly small.

Hence attempting to calculate the average (1) by sampling all of phase space with even probability, one will end up spending most of the time in points $\mathbf{\Gamma}$ with extremely little weight on the average.

What comes to help is **importance sampling,** a central concept in Monte Carlo integration of functions. The basic concept is as follows. We want to integrate some function $f(\mathbf{x})$ in M-dimensional space over some M-dimensional region $V^{(M)}$. By picking N points at random in the region $V^{(M)}$, one can estimate the integral as

$$I \approx V^{(M)} \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \tag{2}$$

where the $\mathbf{x}_i$ are random numbers uniformly distributed in the interval. This may, however, be quite inefficient if f is strongly peaked. If that is the case, one can speed up the integration dramatically by introducing some function $g(x)$ which has a similar peaked shape as $f(\mathbf{x})$. Then by generating random numbers $\mathbf{x}_i^{(g)}$ which are distributed according to $g(\mathbf{x})$, one can calculate the same integral as

$$I = V^{(M)} \frac{1}{N} \sum_{\substack{i=1 \\ \mathbf{x}_i \text{ chosen from g}(\mathbf{x}\ )}}^{N} \frac{f(\mathbf{x}_i^{(g)}))}{g(\mathbf{x}_i^{(g)})} \tag{3}$$

For thermodynamic MC simulation, we saw that large values of E are extremely improbable, which is the same thing as saying that our function is strongly peaked around small values of E. Hence the basic motivation of using importance sampling is directly valid here.

To apply importance sampling here means that the two sums in expression(1) are both calculated by importance sampling using some weight function $w(\mathbf{\Gamma})$,

$$\langle \mathcal{A} \rangle_{\text{ens}} = \frac{\displaystyle\sum_{\substack{\mathbf{\Gamma} \\ \text{chosen from} \\ \text{weight function } w}} A(\mathbf{\Gamma}) \rho_{\text{ens}}(\mathbf{\Gamma}) / w(\mathbf{\Gamma})}{\displaystyle\sum_{\substack{\mathbf{\Gamma} \\ \text{chosen from} \\ \text{weight function } w}} \rho_{\text{ens}}(\mathbf{\Gamma}) / w(\mathbf{\Gamma})} \tag{4}$$

In the general case of MC integration, one does not know how to choose w.

But now we know the weight of our points: it is just ρ_{ens} ! So we can choose

$$w(\mathbf{\Gamma}) = \rho_{\text{ens}}(\mathbf{\Gamma})$$

and obtain the simple result:

$$\langle \mathcal{A} \rangle_{\text{ens}} = \frac{\displaystyle\sum_{\substack{\mathbf{\Gamma} \\ \text{chosen from} \\ \text{weight function } \rho}} A(\mathbf{\Gamma})}{N_{\text{MCsteps}}} \tag{5}$$

The crucial remaining question is then how one can choose points weighted according to the ensemble ρ. This depends on the ensemble, and hence is discussed separately for each ensemble.

13.1 Canonical (NVT) ensemble

13.1.1 Basic properties of ensemble

In the canonical (NVT) ensemble the number of particles, system volume and temperature is conserved. The total energy is given by the Hamiltonian

$$H(\mathbf{\Gamma}) = K(\mathbf{p}) + V(\mathbf{r})$$

where K is the total kinetic energy and V the total potential energy of the system. Here we assume the particles do not have any internal energy I; if they had, that would be added to the sum, and treated separately (see e.g. Mandl ch. 7.8). In the usual classical and semiclassical formulations atoms do not have internal energy terms, but molecules have, in the form of rotational and vibrational degrees of freedom.

The weight function (density) ρ of the canonical ensemble follows a Boltzmann distribution,

$$\rho \propto e^{-H(\mathbf{\Gamma})/kT}$$

and the partition function is, for an atomic system in quasi-classical form,

$$Q_{NVT} = \frac{1}{N!}\frac{1}{h^{3N}} \int d\mathbf{r} d\mathbf{p} \; e^{-H(\mathbf{r},\mathbf{p})/kT}$$

where N is the number of particles, and h Planck's constant. The reason we mention the partition function is its central role in thermodynamics: most other important quantities can be derived from it.

Since $H(\mathbf{\Gamma}) = K(\mathbf{p}) + V(\mathbf{r})$ the partition function factorizes into a product of kinetic (ideal gas) and potential (excess) parts:

$$Q_{NVT} = \frac{1}{N!}\frac{1}{h^{3N}} \int d\mathbf{p} e^{-K(\mathbf{p})/kT} \int d\mathbf{r} e^{-V(\mathbf{r})/kT}$$

This can be written as a product of the ideal gas contribution and the excess contribution as

$$Q_{NVT} = Q_{NVT}^{\mathrm{id}} Z_{NVT}$$

where

$$Z_{NVT} = \int d\mathbf{r} e^{-V(\mathbf{r})/kT}$$

is the so called configuration integral.

The thermodynamic function of the system is the Helmholtz free energy.

$$A = -kT \ln Q_{NVT}$$

which can also be separated:

$$A = -kT \ln(Q_{NVT}^{\mathrm{id}} Z_{NVT}) = -kT \ln Q_{NVT}^{\mathrm{id}} - kT \ln Z_{NVT}$$

The other quantities of the ensemble can be derived from A. Hence this separation is of crucial importance: the first part,

$$kT \ln Q_{NVT}^{\mathrm{id}}$$

is just the term for an ideal gas, the properties of which can be found in any statistical physics textbooks.

Hence to treat the NVT ensemble for interacting particles, we can focus our effort on calculating the potential-energy dependent part Z_{NVT}, then add the known ideal gas properties afterwards.

For MC simulations this means that we can use simulation schemes which do not have to treat atom velocities at all, only the potential energy. Next we present the Metropolis method, which does exactly this.

13.1.2 The Metropolis method

[Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, J. Chem. Phys. 21 (1953) 1087-; Allen-Tildesley]

The original MC simulation scheme for thermodynamics is the famous Metropolis method. Although presented already in 1953[1], it still probably is the most widely used MC simulation scheme.

We now present and prove the method almost identically to the original formulation. In the original paper, the authors considered N hard spheres in a 2D system. We will present it in a slightly more general form, for N particles in 3D interacting by an arbitrary potential energy function $V(\mathbf{r})$ which gives the potential energy for each particle i in the system $V_i(\mathbf{r})$. This potential energy can be any of the schemes presented on this course, classical or quantum mechanical.

The weight function of the NVT ensemble is

$$\rho \propto e^{-V/kT}$$

So now we want to do importance sampling of points with this distribution (instead of evenly in V) to obtain averages of the type given in Eq. 5,

$$\langle \mathcal{A} \rangle_{\text{ens}} = \frac{\displaystyle\sum_{\substack{\mathbf{\Gamma} \\ \text{chosen from} \\ \text{weight function } \rho}} A(\mathbf{\Gamma})}{N_{\text{MCsteps}}}$$

I.e. we want to choose sampled states with a probability proportional to

$$e^{-V/kT} \tag{6}$$

In practise, one wants to move one particle at a time in the system. Then it becomes easier to work with the energy *difference* ΔE between the new and the previous state, rather than the energy of the whole system. This is done in the Metropolis algorithm in a way which leads to the desired probability of sampling states (6). We first present the algorithm, then prove that this leads to (6).

We consider a system with N particles at positions $\mathbf{r}_i$.

[1]They actually just this year, 2003, celebrated the 50th anniversary of the algorithm in Los Alamos

0 a° Place the particles in some initial configuration $\mathbf{r}_i$.

0 b° Choose a maximum displacement vector (d_x, d_y, d_z)

0 c° Set the number of Monte Carlo steps $n_{MCS} = 0$ and $A_{\text{sum}} = 0$

1° Choose a particle i at random among the N particles

2° Calculate the energy of the system before the transition $E_b = V(\mathbf{r})$

3° Generate three uniform random numbers u_1, u_2, u_3 **between -1 and 1**

4° Displace atom i by $\Delta\mathbf{r} = (u_1 d_x, u_2 d_y, u_3 d_z)$: $\mathbf{r}_i = \mathbf{r}_i + \Delta\mathbf{r}$

5° Calculate the energy of the system after the transition $E_a = V(\mathbf{r})$

6° Calculate $\Delta E = E_a - E_b$, then :

- **7**° If $\Delta E \leq 0$ accept the state

- **8**° If $\Delta E > 0$:

  - **8 a**° Generate a random number u between 0 and 1

  - **8 b**° Accept the state only if $u < e^{-\Delta E/kT}$

9° If the state is rejected, return to the previous state: $\mathbf{r}_i = \mathbf{r}_i - \Delta\mathbf{r}$

10° Sum up the desired physical property A for positions $\mathbf{r}$: $A_{\text{sum}} = A_{\text{sum}} + A(\mathbf{r})$

11° Set $n_{\text{MCS}} = n_{\text{MCS}} + 1$

12° If $n_{\text{MCS}} < n_{\max}$ return to step 1

13° Calculate and print out the desired average $\dfrac{A_{\text{sum}}}{n_{\max}}$

We can now note several things directly related to the algorithm:

1. The physical properties should be calculated at each MC step regardless of whether the jump is accepted or rejected.

2. The atom velocities or forces between atoms are not needed anywhere. Indeed, it is not even clear how velocities could be defined in this scheme.

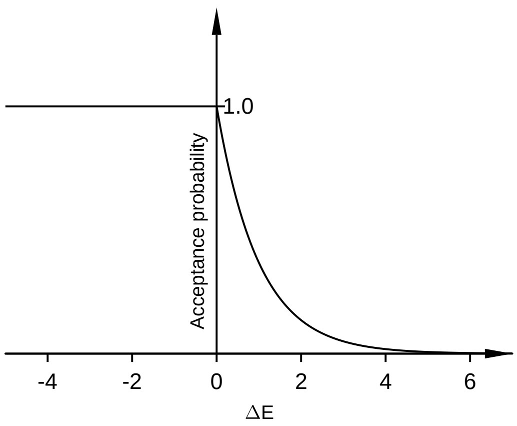
3. There is no time scale involved here. We do count the number of Monte Carlo steps n_{MCS} but it is not in any way clear whether these correspond to any real time scale.

4. The only steps really specific to an atomic system are 3° , 4° and 9° . The energy could actually be any variable measuring the state of a system, steps 3° and 4° could be replaced by any kind of a trial, and step 9° by any kind of a return-back operation. This gives a quite general algorithm which can be used for instance for minimization in many contexts which have nothing to do with atoms.

5. The vector **d** is clearly a free parameter. Its choice will affect how many of the trials are accepted or rejected. A general rule of thumb is that about half of the jumps should be accepted for optimally efficient probing of phase space.

6. For any given system one can then find a good choice of **d** more or less by trial and error. Or one can easily devise an adaptive scheme for this: if the number of accepted jumps over the last, say, 1000 steps is much below 0.5, decrease **d** , and vice versa.

The basic method of acceptance can be illustrated by the following graph, giving the acceptance rate as a function of ΔE:



13.1.2.1 Proof that this produces the right distribution of states

Although the algorithm clearly seems related to the desired distribution of states

$$e^{-V/kT}$$

it is not immediately obvious that it really produces states in this distribution. Hence we will now prove it; the proof is directly from the original paper.

We assume the ergodic hypothesis to be true, i.e. that all states can be reached. If the trial distance **d** is large enough, any atom can jump anywhere in the system. Since in this case the list of possible points in phase space only involved the atom positions, it thus seems quite likely the ergodic hypothesis really holds now.

Consider now a large number of NVT systems. We consider again discrete systems with a finite number of possible states. Let ν_r be the number of systems which are in a specific state r, which has an energy E_r. What we must prove is that after many moves in each system, the distribution of energies in the different systems tends towards the Boltzmann distribution, i.e. that

$$\nu_r \propto e^{-E_r/kT} \tag{7}$$

Now let's make a move in all the systems. The probability that the transition would go from state r to state s *before* we account for the $\exp(-\Delta E/kT)$ factor is called P_{rs}. Since all states and transitions of the type described above are equal,

$$P_{rs} = P_{sr}$$

Now assume $E_r > E_s$. Then the number of systems moving from state r to state s will simply be

$$\nu_r P_{rs}$$

since all transitions to lower energies are allowed in the Metropolis scheme. But the number of systems doing the opposite transition $s \to r$ will be

$$\nu_s P_{sr} e^{-(E_r - E_s)/kT}$$

Now the net number of systems $M_{s\to r}$ moving from s to r is

$$M_{s\to r} = \nu_s P_{sr} e^{-(E_r-E_s)/kT} - \nu_r P_{rs} = P_{rs}(\nu_s e^{-(E_r-E_s)/kT} - \nu_r) = P_{rs}(\nu_s \frac{e^{-E_r/kT}}{e^{-E_s/kT}} - \nu_r)$$

since $P_{sr} = P_{rs}$. From this we see that if

$$\nu_s \frac{e^{-E_r/kT}}{e^{-E_s/kT}} < \nu_r$$

then the net movement $M_{s\to r}$ is negative, i.e. more systems move from r to s. This clause can be rewritten by dividing with ν_s. This says then that if

$$1^\circ \qquad \frac{\nu_r}{\nu_s} > \frac{e^{-E_r/kT}}{e^{-E_s/kT}}: \qquad \text{Net movement is from } r \text{ to } s \tag{8}$$

$$2^\circ \qquad \frac{\nu_r}{\nu_s} < \frac{e^{-E_r/kT}}{e^{-E_s/kT}}: \qquad \text{Net movement is from } s \text{ to } r \tag{9}$$

What does this mean? Compare this with the desired result, Eq. (7). We see that if this condition is already fulfilled for both r and s, there is no net movement in any direction, i.e. the steady state is indeed given by Eq. (7), as it should.

On the other hand, let us imagine Eq. (7) is not fulfilled. Then if condition 1° is fulfilled, there are too many states in r, and the algorithm drives the system from r towards s. Conversely, if condition 2° is fulfilled, there are too many states in s, and the system is driven from s towards r. In either case, the result is that the system is driven towards the correct distribution, Eq. (7). Since we assume ergodicity, it is clear the final state will eventually be reached Hence this completes the proof.

Note that from this proof we have no idea how many steps it would actually take to reach the equilibrium distribution.

13.1.2.2 Comments on proof

We can restate the central step in the Metropolis algorithm by saying that the transition probability $r \to s$ is

$$W_{r\to s} = \min(1, e^{-\Delta E/kT})$$

because if $\Delta E < 0$, the exponential becomes larger than 1. It turns out that the proof above is valid even for other forms of $W_{r\to s}$; any kind of transition which fulfills the so called **detailed balance condition**

$$W_{r\to s} e^{-E_r/kT} = W_{s\to r} e^{-E_s/kT}$$

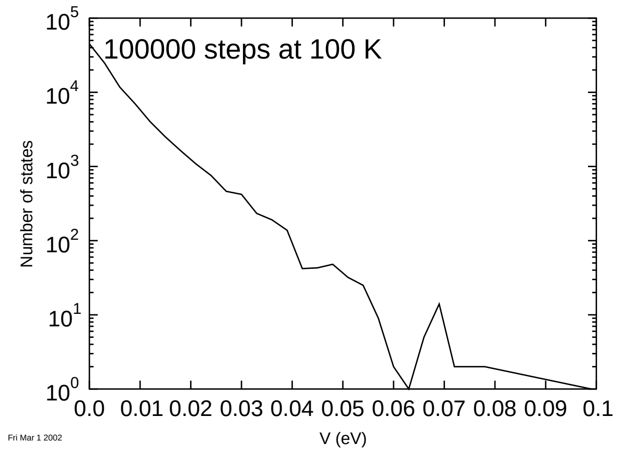
will produce the Boltzmann distribution. This is quite easy to see by comparison with the proof given above.

13.1.3 Example: single particle

As a first, very basic test I implemented the Metropolis algorithm on a single particle, interacting with a simple harmonic potential:

$$V = 5x^2$$

and simulated the movement of this particle with the Metropolis algorithm (this way we do not have to worry about the density of states). The resulting distribution is



which is a nice exponential distribution to a pretty good precision.

13.1.4 Example: Cu with Morse potential

To obtain a real-life example of the Metropolis MC method, I simulated Cu with the Morse potential, using a cutoff of 7.0 Å. With this potential and cutoff the equilibrium lattice constant at 0 K is $a = 3.62538$ and the cohesive (potential) energy is -3.49903 eV/atom.

In coding this, I did one trick for speedup. The energy of the system is calculated with a subroutine, which can be called so that it either calculates the energy of a single atom i, or the total energy of the system. In the Metropolis algorithm when applied for pair potentials, it is enough to calculate the energy of a single atom i to get the change in energy. In my code, this is implemented such that the energy of the atom i is calculated both before and after the trial change in a position.

An even more efficient implementation would be to keep track of the potential energy of each atom, correcting it after every accepted trial. This way one would have to calculate the energy on atom i only once every trial. But this would make the code less clear, and hence I chose not to implement this.

Many of the tricks presented previously on this course (neighbour list and link-cell method) could be used to speed up the present implementation if the number of atoms is large.

Here is the central part of the code.

```
do
   ! Trial jump
   ! ---------- Select atom ----------------------------
   i=int(grnd()*N)+1; if (i>N) i=N;
   ! ---------- Get old energy for atom i --------------
   call energy(atom,N,size,Eiold,i)
   ! ---------- Move it randomly -----------------------
   dx=(1.0-2.0*grnd())*djump;
   dy=(1.0-2.0*grnd())*djump;
   dz=(1.0-2.0*grnd())*djump;
   atom(i)%x=atom(i)%x+dx
   atom(i)%y=atom(i)%y+dy
   atom(i)%z=atom(i)%z+dz
   if (atom(i)%x >= size/2) atom(i)%x=atom(i)%x-size
   if (atom(i)%x < -size/2) atom(i)%x=atom(i)%x+size
   if (atom(i)%y >= size/2) atom(i)%y=atom(i)%y-size
   if (atom(i)%y < -size/2) atom(i)%y=atom(i)%y+size
   if (atom(i)%z >= size/2) atom(i)%z=atom(i)%z-size
   if (atom(i)%z < -size/2) atom(i)%z=atom(i)%z+size

   ! ---------- Get new energy for atom i --------------
   call energy(atom,N,size,Einew,i)
   accept=.false.
   if (Einew <= Eiold) then
      accept=.true.
   else
      if (grnd() < exp(-(Einew-Eiold)*e/(kB*T))) accept=.true.
   endif
   if (accept) then
      isteplastsuccess=istep
      nsuccess=nsuccess+1
      ! Follow evolution of total energy
      Etot=Etot-Eiold+Einew
   else
      atom(i)%x=atom(i)%x-dx
      atom(i)%y=atom(i)%y-dy
      atom(i)%z=atom(i)%z-dz
      if (atom(i)%x >= size/2) atom(i)%x=atom(i)%x-size
      if (atom(i)%x < -size/2) atom(i)%x=atom(i)%x+size
```

```
      if (atom(i)%y >= size/2) atom(i)%y=atom(i)%y-size
      if (atom(i)%y < -size/2) atom(i)%y=atom(i)%y+size
      if (atom(i)%z >= size/2) atom(i)%z=atom(i)%z-size
      if (atom(i)%z < -size/2) atom(i)%z=atom(i)%z+size
      nfail=nfail+1
   endif
   istep=istep+1

   ! ---------- Collect physical results ---------------
   if (istep>istepcollect) then
      Esum=Esum+Etot
      Eave=Esum/(N*(istep-istepcollect))
   endif

   ! ---------- Print out results every now and then ---
   if (mod(istep,10000)==1) then
      ! Get total energy from scratch to prevent numerical
      ! drift
      call energy(atom,N,size,Etot,0)

      print *,"E",istep,1.0*nsuccess/istep,Etot/N,Eave
   endif

   if (istep>= isteplastsuccess+Nfailmax) exit
   if (istep>= istepmax) exit
enddo
```

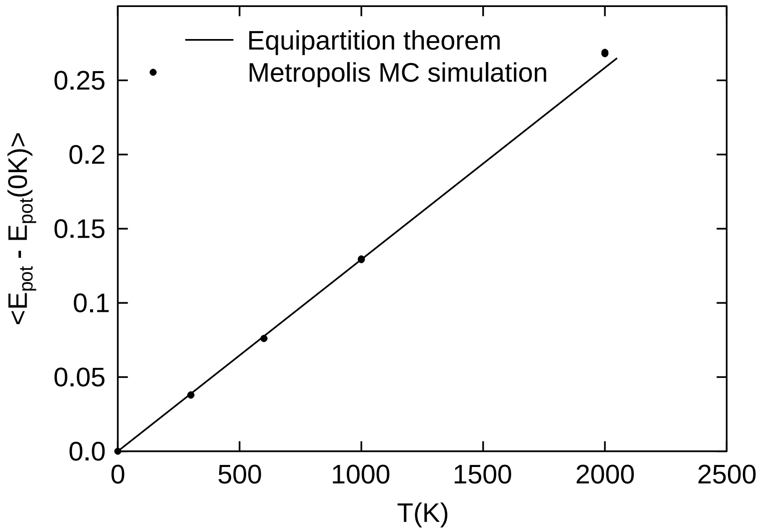
As a first test of the code (after trivial debugging), I simulated Cu starting from the perfect FCC structure at different temperatures, and calculated the average potential energy at each temperature. This tests the simulation, since the energy equipartition theorem says that in a classical system at constant temperature, the average potential energy should be $\frac{1}{2}kT$ per degree of freedom. Now we have exactly 3 degrees of freedom per atom (x, y and z), so the average potential energy per atom, compared to the result at 0 K, should just be $\frac{3}{2}kT$

The simulations gave the following result (`djump` is the maximum jump distance in the Metropolis model)

T	djump	Eave	Eave-Eave(0K)	Equipartition prediction
----	-----	----	-------------	---------
0	0.2	-3.4990306	0.0	0.0
300	0.2	-3.4612282	0.03780	0.038778
300	0.2	-3.4609813	0.0380493	0.038778
600	0.2	-3.4229604	0.07607	0.07755
600	0.2	-3.4230638	0.075967	0.07755
1000	0.2	-3.3699146	0.129116	0.12926
1000	0.2	-3.3693646	0.129666	0.12926

2000	0.2	-3.2305089	0.268521	0.25852
2000	0.2	-3.2301234	0.268907	0.25852
2000	0.4	-3.2310927	0.267937	0.25852
2000	0.4	-3.2309152	0.268115	0.25852

The comparison is better illustrated by a graph:



which clearly shows that up to 1000 K, the theoretical and simulation prediction are in excellent agreement. This supports the theoretical prediction that the Metropolis method indeed produces the NVT ensemble.

At 2000 K there is some deviation. I did not analyze what could be the reason to this, but there are several possibilities. One is that the artificial cutoff at 7 Å causes distortions in the data. Another is that at this temperature defects starts forming (2000 K is actually more than the experimental melting point of Cu, but not more than the melting point of the crude Morse model of Cu). A third is that because the potential is not harmonic, anharmonic effects start to play in at high temperatures.

After this encouraging result, I did a much more ambitious test. I generated 256 random atom positions for a cell with a size which corresponds to $4 \times 4 \times 4$

unit cells in Cu. So the equilibrium structure of this system is the perfect FCC at all temperatures below melting. So the Metropolis MC method should in principle find the fcc structure, and then return the same values for `Eave` as in the table above, for each T.

But in the simulations I never managed to get to the correct ground-state structure from the random positions, despite trying fairly hard. Here are some results:

T	djump	Nsteps	fraction of failures	Eave
2000	0.2	1e6	0.412087	-3.061507
1000	0.2	1e6	0.268201	-3.210508
1000	0.4	1e6	6.099085E-02	-3.202355
600	0.2	1e6	0.172835	-3.269553
600	0.2	3e6	0.171711	-3.284650
300	0.2	3e6	7.716486E-02	-3.304991
300	0.1	3e6	0.311042	-3.316886
100	0.1	3e6	0.110436	-3.334477
100	0.03	10e6	0.577190	-3.347841

Comparison with the `Eave` results in the table above shows that we always are above the desired result, by some ~ 0.2 eV/atom. Experimentally speaking, this is huge; the stored potential energy would, if suddenly converted to kinetic energy, heat the sample by some 1500 K! In real Cu, this would melt the whole sample.

This shows that although the system almost certainly is ergodic, there are large barriers which may prevent sampling all of phase space, and hence leading to completely wrong results.

And it is especially important to realize that it was only our good physical understanding of this system which allowed us to know that we are getting the wrong results: the latter group of simulations, starting from the random positions, do not directly reveal that we do not obtain the right structure. Although the fact that different choices of `Nsteps` and `djump` lead to different results, should have given us a clue that something is fishy.

13.2 Microcanonical (NVE) ensemble

13.2.1 Basic properties of the ensemble

[Allen-Tildesley ch 2.2, Mandl]

In the microcanonical (NVE) ensemble the number of particles, system volume and total energy is conserved. This corresponds to a completely closed system which does not interact in any way with the environment, and is in a container of fixed volume V.

Again we will for simplicity neglect internal degrees of freedom. Then the system energy will again be a sum of kinetic and potential energies. Now since the total energy E must be conserved, the criterion will simply be that the total Hamiltonian

$$H(\mathbf{\Gamma}) = K(\mathbf{\Gamma}) + V(\mathbf{\Gamma}) = \text{constant} = E$$

which means that not all, but only those states in phase space $\mathbf{\Gamma}$ that conserve the total energy are allowed. This can also be stated so that the probability density of the ensemble is

$$\delta(H(\mathbf{\Gamma}) - E)$$

where the δ is the Kronecker delta for a discrete system, and the Dirac delta function for a continuous one.

The partition function can be written

$$Q_{NVE} = \frac{1}{N!}\frac{1}{h^{3N}} \int d\mathbf{r}d\mathbf{p}\ \delta(H(\mathbf{\Gamma}) - E)$$

where N is the number of particles, and h Planck's constant.

As we have seen, simulating this ensemble with MD is straightforward, since direct numerical solution of Newton's equations of motion conserves energy. For MC we need to find some algorithm which will produce the NVE ensemble. In the next subsection I present such a method.

13.2.2 The demon algorithm

[G+T ch. 16-3; Creutz, Phys. Rev. Lett. 50 (1983) 1411]

We rewrite the microcanonical partition function as

$$Q_{NVE} = \frac{1}{N!}\frac{1}{h^{3N}} \int d\mathbf{r}d\mathbf{p}\ \delta(K(\mathbf{p}) + V(\mathbf{r}) - E)$$

which just emphasizes that the total energy has potential V and kinetic energy K parts. For MC simulations similar to the Metropolis type, we have the problem that we do not have any treatment of the atom kinetic energies, and hence can not calculate $K(\mathbf{p})$ explicitly.

In the demon method devised by Creutz in 1993, the main idea is to replace the kinetic energy of all N atoms,

$$K(\mathbf{p}) = \sum_{i=1}^{N} \frac{p^2}{2m}$$

with a single "demon energy" E_D which simulates the true sum, i.e.

$$Q_{NVE} = \frac{1}{N!}\frac{1}{h^{3N}} \int d\mathbf{r}d\mathbf{p}\ \delta(E_D + V(\mathbf{r}) - E)$$

Note, however, that E_D and $K(\mathbf{p})$ are not directly comparable. The demon runs around the system, exchanging energy with the particles in it either in sequential or random order. But there has to be some restriction on how it can do it, to prevent it from sucking up all the energy in the system. The restriction is simply that the demon energy should always be positive (which is quite natural, as the system kinetic energy of course always stays positive).

The algorithm can be written as follows. The parts which differ from the Metropolis algorithm are **written in bold**.

0 a° Place the particles in some initial configuration $\mathbf{r}_i$.

0 b° Choose a maximum displacement vector (d_x, d_y, d_z)

0 c° Set the number of Monte Carlo steps $n_{MCS} = 0$ and $A_{\text{sum}} = 0$

0 d° **Set the demon energy** $E_D = 0$.

1° Choose a particle i at random among the N particles

2° Calculate the energy of the system before the transition $E_b = V(\mathbf{r})$

3° Generate three uniform random numbers u_1, u_2, u_3 between -1 and 1

4° Displace atom i by $\Delta\mathbf{r} = (u_1 d_x, u_2 d_y, u_3 d_z)$: $\mathbf{r}_i = \mathbf{r}_i + \Delta\mathbf{r}$

5° Calculate the energy of the system after the transition $E_a = V(\mathbf{r})$

6° Calculate $\Delta E = E_a - E_b$, then :

- **7°** If $\Delta E \leq 0$ accept the state and **give the energy** ΔE **to the demon,** $E_D = E_D - \Delta E$
- **8°** If $\Delta E > 0$:
  - **8 a°** **If** $E_D \geq \Delta E$ **let the demon give the necessary energy to the system,** $E_D = E_D - \Delta E$**, and accept the configuration**
  - **8 b°** **Otherwise the configuration is rejected.**

9° If the state is rejected, return to the previous state: $\mathbf{r}_i = \mathbf{r}_i - \Delta\mathbf{r}$

10° Sum up the desired physical property A for positions $\mathbf{r}$: $A_{\text{sum}} = A_{\text{sum}} + A(\mathbf{r})$

11° Set $n_{\text{MCS}} = n_{\text{MCS}} + 1$

12° If $n_{\text{MCS}} < n_{\text{max}}$ return to step 1

13° Calculate and print out the desired average $\dfrac{A_{\text{sum}}}{n_{\text{max}}}$

Note that it is only 3 steps which differ from the Metropolis approach. So if we have a Metropolis MC (NVT) code for a given system, it is very easy to turn it into a demon (NVE) code. Similarly, many of the basic features of the Metropolis method, such as how the physical quantities are calculated, and that there are no atom velocities, are directly valid here as well.

We can now note a few other things directly related to this particular algorithm:

1. The total energy (demon E_D + atom potential energy E_{pot}) obviously always stays constant. But because the demon is a single degree of freedom, compared to the many atom degrees of freedom, the fluctuations in the atom potential energy are only of order $1/N$.

2. The energy of the demon will have a Boltzmann distribution,

$$P(E_D) \propto e^{-E_D/kT}$$

This observation allows us to define a temperature in the system as

$$kT = \langle E_D \rangle$$

where $\langle E_D \rangle$ is the long-term average of the demon energy.

3. The energy of the demon will be very small compared to the energy of the whole system, for large numbers of atoms.

This algorithm can be converted to the Metropolis algorithm at least in two ways. If the demon is connected to a heat bath at each time step just before step **8°** , i.e. given a new energy with a Boltzmann weight $\exp(-E/kT)$, then the algorithm will identically become the Metropolis one.

Also, if one releases a large number of demons rather than only one, their total energy will become appreciable, and with increasing numbers of demons one will eventually reach the Metropolis (NVT) distribution. This means one can smoothly move between the NVE and NVT ensembles by modifying the number of demons!

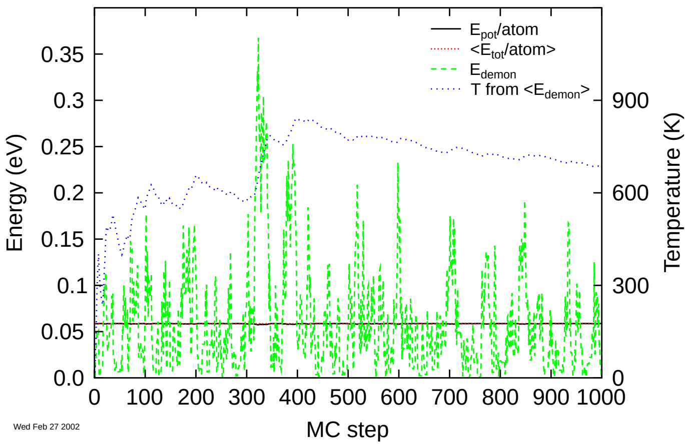
13.2.2.1 Example: Cu again

As an example, I used the demon algorithm to simulate the same 256 atom FCC Morse Cu system used to test the Metropolis approach. Rewriting the Metropolis code to a "demon" code only took a few minutes; the main routine is identical except for the way in which the trials are tested for. The central section of the code is

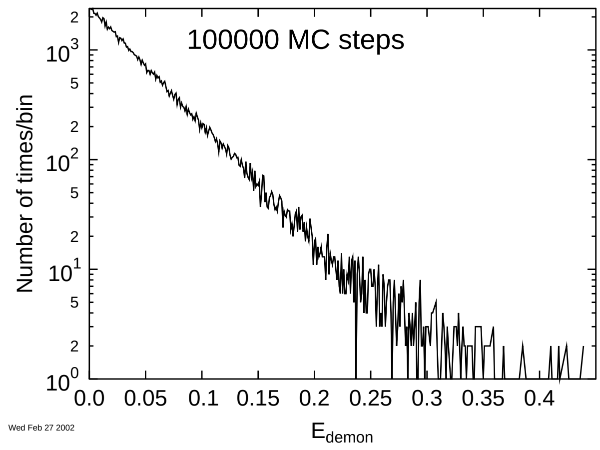
```
do
   ! ----------------- Trial jump ---------------------
   (OMITTED, SEE METROPOLIS MC CODE)
   ! ---------- Get new energy for trial atom i ---------
   call energy(atom,N,size,Einew,i)

   ! ------------- Demon algorithm --------------------
   accept=.false.
   DeltaE=Einew-Eiold
   if (DeltaE <= 0.0) then
      accept=.true.
      Edemon=Edemon-DeltaE
   else
      if (Edemon > DeltaE) then
         accept=.true.
         Edemon=Edemon-DeltaE
      else
         accept=.false.
      endif
   endif
   if (accept) then
      isteplastsuccess=istep
      nsuccess=nsuccess+1
      ! Follow evolution of total energy
      Etot=Etot-Eiold+Einew
   else
      (OMITTED, SEE METROPOLIS MC CODE)
   endif
   istep=istep+1

   ! ----------  Collect physical results -------------
   if (istep>istepcollect) then
      Esum=Esum+(Etot-Eorig)
      Eave=Esum/(N*(istep-istepcollect))
      Edemonsum=Edemonsum+Edemon
      Edemonave=Edemonsum/(istep-istepcollect)
   endif

   ! Estimate temperature assuming Ed is Boltzmann distributed
   T=Edemonave*e/kB
   print *,"E",istep,1.0*nsuccess/istep,(Etot-Eorig)/N,Eave,Edemon,T

   ! --------- Print out results every now and then ----------
```

```
    (OMITTED, SEE METROPOLIS MC CODE)
enddo
```

I ran this for a while, and printed out $E_{\rm pot}$ (calculated above the ground state), $\langle E_{\rm pot}\rangle$ $E_{\rm demon}$, and the temperature calculated using $k_B T = \langle E_{\rm demon}\rangle$. This gave the following result:



We see that the demon energy fluctuates heavily, as it should. The potential energy per atom, on the other hand, is quite constant, as it should be by the prediction that the energy fluctuations in the atomic system are only of the order of $1/N$. If we make a statistics out of the demon energy, we see that it indeed has a Boltzmann distribution:



Note that we should not interpret

$$k_B T_{\text{instantaneous}} = E_{\text{demon,instantaneous}} \qquad \text{(not good)}$$

as the instantaneous temperature of the system, since in a real system the temperature does not fluctuate nearly as heavily as the demon energy. It is only the interpretation of the average demon energy $\langle E_{\text{demon}} \rangle$ as corresponding to temperature that makes sense!

13.3 Isobaric-isothermal (NPT) ensemble

[Allen-Tildesley 4.5, Gould-Tobochnik 17.9]

The way to simulate the NPT ensemble is in principle quite similar to that in the NVT ensemble, the main difference being that the volume is not held constant, and we need a 'volume' trial as well.

To avoid confusion in the notation, we now use the following symbols:

$$E \quad \text{for the Hamiltonian (previously } H\text{)}$$

$$U \quad \text{for the potential energy (previously } V\text{)}$$

The probability density for this ensemble is

$$e^{-(E+PV)/kT}$$

i.e. we also have the pressure term determining what pressure we are working on, and the volume V has been added to the set of state points $\mathbf{\Gamma}$. The characteristic thermodynamic quantity is the enthalpy H

$$H = E + PV$$

The partition function is

$$Q_{NPT} = \frac{1}{N!}\frac{1}{h^{3N}}\frac{1}{V_0}\int dV \int d\mathbf{r}d\mathbf{p}\; e^{-(E+PV)/kT}$$

and by leaving out the kinetic energy terms we get the configuration integral

$$Z_{NPT} = \int dV e^{-PV/kT}\int d\mathbf{r}e^{-U/KT}$$

The average of a quantity A is obtained using

$$\langle A\rangle_{NPT} = \frac{\int_0^\infty dV e^{-PV/kT}V^N \int d\mathbf{s}A(\mathbf{s})e^{-U(\mathbf{s})/kT}}{Z_{NPT}}$$

Here an important change to the usual setup has been made: instead of the normal coordinates $\mathbf{r}$ ranging from 0 to L (or $-L/2$ to $L/2$) in each dimension (L is the system size in each dimension), we use unitless position coordinates

$$\mathbf{s} = \frac{\mathbf{r}}{L}$$

which range from 0 to 1, or -1/2 to 1/2. This is because when the system volume keeps changing, the coordinates $\mathbf{r}$ would need to be scaled to the new size every time the size is changed. But the coordinates $\mathbf{s}$ stay unchanged!

Now the distribution to be approached is also different. Since the equation to calculate an average for the quantity A has a factor of

$$e^{-PV/kT}V^N = e^{-PV/kT}e^{\ln V^N} = e^{-PV/kT+N\ln V}$$

the distribution to be approached is

$$e^{-U(\mathbf{s})/kT-PV/kT+N\ln V} \tag{10}$$

The way to simulate the NPT ensemble is to use a Metropolis scheme, but instead of just the trial on moving an atom, there is an additional trial on changing the volume. This trial is conceptually identical to the moving of an atom:

3 V° Generate random number u_V between -1 and 1

4 V° Change system size V by $\Delta V = u_V d_V$: $V_t = V + \Delta V$

where d_V is some predetermined maximum volume change.

Then we evaluate the change in the exponent of Eq. 10,

$$\Delta W = \Delta U(\mathbf{s}) + P(V_t - V) - NkT \ln \frac{V_t}{V}$$

and accept the trial as follows:

7 V° If $\Delta W \leq 0$ accept the state

8 V° If $\Delta W > 0$:

8 a V° Generate a random number u between 0 and 1

8 b V° Accept the state only if $u < e^{-\Delta W/kT}$

A few things can be noted here:

1. Calculating the new $U(\mathbf{s})$ requires calculating all interactions from scratch. Hence the volume trial is much slower than the atom trial, and it is not worth attempting it at every step.

2. P is the desired pressure. Note that we do not necessarily need to be able to calculate the pressure; we can in principle just trust that the algorithm should lead to the right pressure.

3. But in practice it is almost certainly healthy to be able to calculate the pressure to check that we are coming to the right answer. This can be either done exactly by calculating the virial of the system, or approximately by numerically evaluating $\frac{\partial U}{\partial V}$. Calculating the virial requires information on the forces between atoms, which are not otherwise needed in MC, so doing the numerical estimate is much easier to implement, and in my experience often accurate enough.

But of course if a potential routine with forces is already available, it is worth using it.

13.4 Grand canonical (μVT) ensemble

[Allen-Tildesley 4.6, Gould-Tobochnik 17.9]

In the grand canonical ensemble, a system is in contact with a heat bath with which it also can exchange particles. The probability density for this ensemble is

$$e^{-(E-\mu N)/kT}$$

where μ is the chemical potential, i.e. the derivative of the free energy with respect to the the number of particles in the system,

$$\mu = \frac{\partial G}{\partial n}$$

If there is only a single component in the system this reduced to the simple form

$$\mu = \frac{G}{N}$$

The partition function is

$$Q_{\mu VT} = \sum_N \frac{1}{N!} \frac{1}{h^{3N}} e^{\mu N/kT} \int d\mathbf{r} d\mathbf{p} \; e^{-E/kT}$$

and the average of a quantity A is obtained using

$$\langle A \rangle_{\mu VT} = \frac{\sum_{N=0}^{\infty} \frac{1}{N!} V^N \left(\frac{e^{\mu/kT}}{\Lambda} \right)^N \int d\mathbf{s} A(\mathbf{s}) e^{-U(\mathbf{s})/kT}}{Q_{\mu VT}}$$

where

$$\Lambda = \sqrt{\frac{h^2}{2\pi mkT}}$$

is the thermal de Broigle wavelength

This is a bit complicated, and on this course we do not have the time to go into the meaning of all the terms. But the important thing now is that the distribution we want to reach is

$$e^{-U(\mathbf{s})/kT + \mu N/kT - N \ln \Lambda^3 - \ln N! + N \ln V} \tag{11}$$

Now there is no single clear way in which we can achieve this. The ordinary particle trial is still as in the Metropolis scheme. But we also need to modify the number of particles in the simulation, to see which average number is the equilibrium one. One way to do this is create another reservoir system of "ghost" particles which move around freely, and sometimes allow particles to switch between the real and ghost system, with a Metropolis scheme utilizing Eq. 11.

Another, and according to Allen-Tildesley the most used scheme is the simpler one, in which there are three different kinds of trials:

a° Displace a particle, as in original Metropolis scheme

b° Destroy a particle

c° Create a new particle in a random position in the system

The criteria for accepting trials b° and c° are as follows. Denote

$$\mu^* = \mu - kT \ln \frac{\Lambda^3}{V}$$

which allows the exponent in 11 to be written as

$$W = \mu^* N/kT + \ln N! + U/kT$$

If W is increased both trials b° and c° are accepted. If W decreases, the change is accepted with probability

$$\text{b, destruction:} \qquad N e^{-(\mu^* + (U_{N+1} - U_N))/kT}$$

$$\text{c, creation:} \qquad \frac{1}{N+1} e^{(\mu^* - (U_{N+1} - U_N))/kT}$$

Since we want to simulate constant T, V and μ, μ^* is a constant input parameter for the simulation.

In practical implementation of this, one would have to consider that when a particle is created or destroyed, empty spaces may be left in the arrays. One could either move particles down to fill these in, but this is rather inefficient since the particle information may be in several different arrays. Another option is to use a separate array `LOCATE` which points to the particles which are alive.

13.5 Calculating thermodynamic quantities

On this course, we have already dealt with how one can obtain thermodynamic quantities in the theory package. All of the equations given there which do not involve velocities can be directly used in MC as well.

Here I just list the equations again, since repetition never hurts.

Total energy E:

$$E = \frac{3}{2}NkT + \langle U \rangle$$

Pressure:

$$PV = NkT + \langle W \rangle$$

where the virial W is obtained from the forces

$$W = -\frac{1}{3}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} \mathbf{r}_{ij}\mathbf{f}_{ij}$$

where $\mathbf{r}_{ij}$ is the distance between atoms i and j and $\mathbf{f}_{ij}$ the force acting between them.

Utilizing the energies, temperatures and pressures several quantities can be calculated with the following basic thermodynamic equalities:

$$\text{constant volume heat capacity} \quad C_V = \left(\frac{\partial E}{\partial T}\right)_V \tag{12}$$

$$\text{constant pressure heat capacity} \quad C_P = \left(\frac{\partial H}{\partial T}\right)_P \tag{13}$$

$$\text{Thermal expansion coefficient} \quad \alpha_P = \frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_P \tag{14}$$

$$\text{Isothermal compressibility} \quad \beta_T = -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_T \tag{15}$$

$$\text{Thermal pressure coefficient} \quad \gamma_V = \left(\frac{\partial P}{\partial T}\right)_V \tag{16}$$

$$\text{Bulk modulus} \quad B = -V\left(\frac{\partial P}{\partial V}\right)_T \tag{17}$$