Physics 115/242
Molecular dynamics project: sample answer

Peter Young

I took $N = 30$ particles coupled with anharmonic springs as discussed in the text. Periodic boundary conditions were used so each particle has two neighbors. I used the position Verlet algorithm with timestep $h = 0.01$. I let the system equilibrate for a time $t_{\rm equil} = 20000$ and then measured the velocity of each particle at integer times during the subsequent period of $t_{\rm meas} = 1000000$. I formed bins of size 0.02 to create the histogram.

The mean square velocity was also computed gave the temperature of the system to be

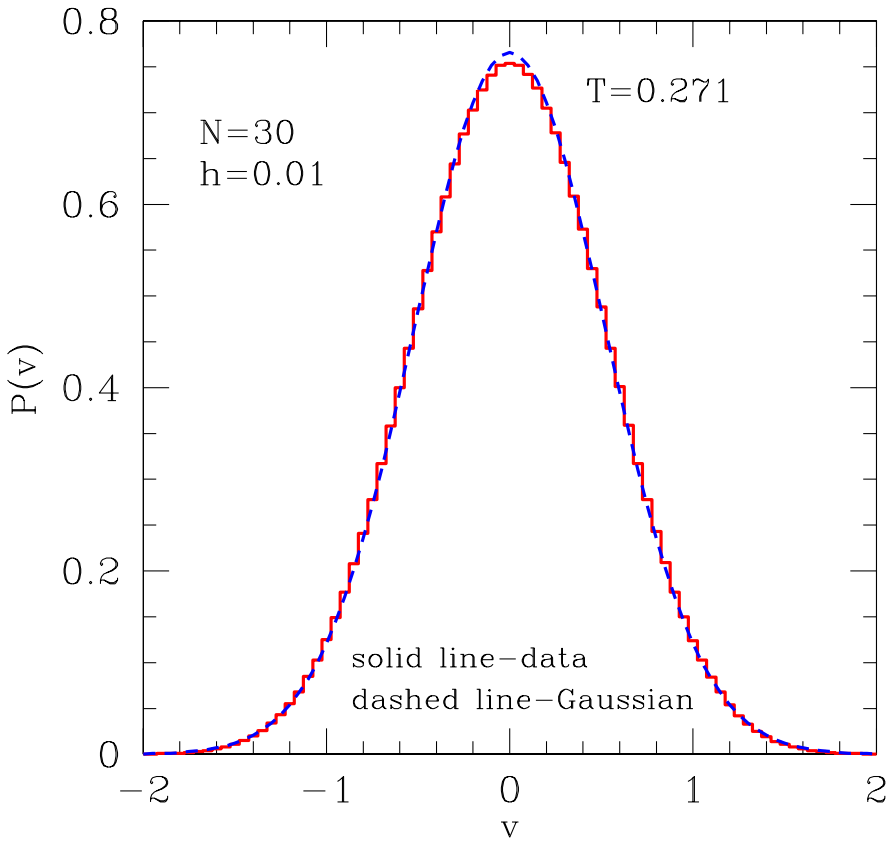
$$\langle v^2 \rangle = T = 0.271. \tag{1}$$

Hence the distribution of velocities should be given by

$$P(v) = \frac{1}{\sqrt{2\pi T}} \exp\left(-\frac{v^2}{2T} \right) \tag{2}$$

with T given by Eq. (1).

The dashed line in the figure below shows the expected distribution of velocities given by Eq. (2). The histogram in the figure is the (normalized) histogram of velocities obtained in the simulation. It is seen that the two agree very well.

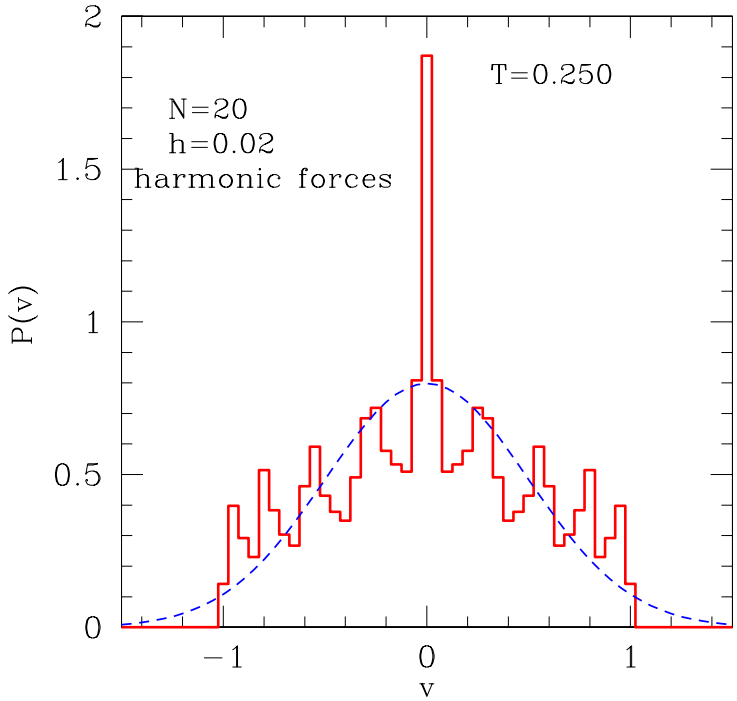


To get good agreement I found it was necessary to use a fairly small time step h in the leapfrog algorithm. With a much larger h I found that the computed distribution was a little too low around $v = 0$.

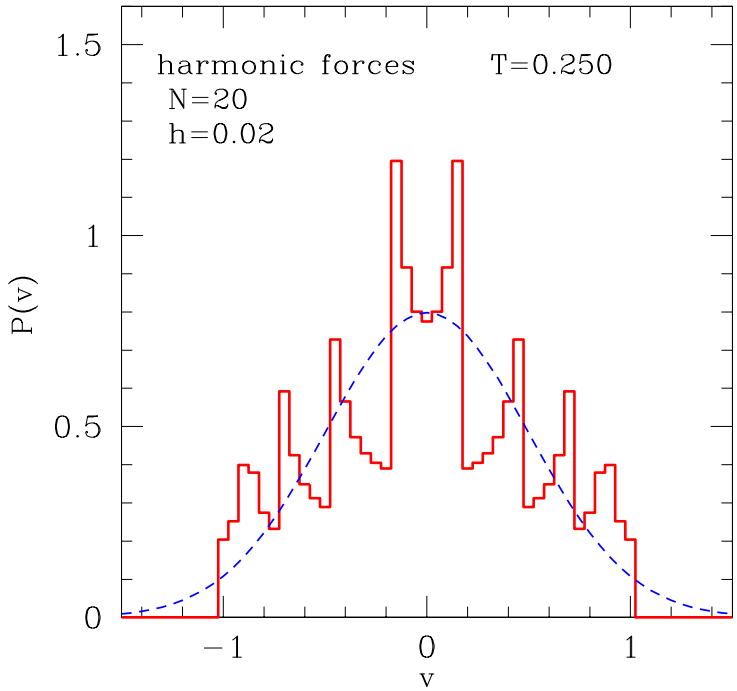
Note that the source code below is divided into subroutines (functions) which do a particular job, for example defining the force function and and integrating one time-step. This makes the program easier to read. It is also easier to debug if there is one piece of code to do one job, which is then called at different points in the program, rather than having copies of the code at each place where the job has to be done. In the latter case, changes have to be made in each copy, which is more error prone.

I then switch off the anharmonic term in the potential (i,e, the part of the force involving the cube of the distance), which I do by changing one line of code, the function f now returns $-y$ rather than $-y - y^2 - y^3$. The temperature is then exactly 0.25 (which indeed I find) and the distribution is not Gaussian, as shown in the figure below.

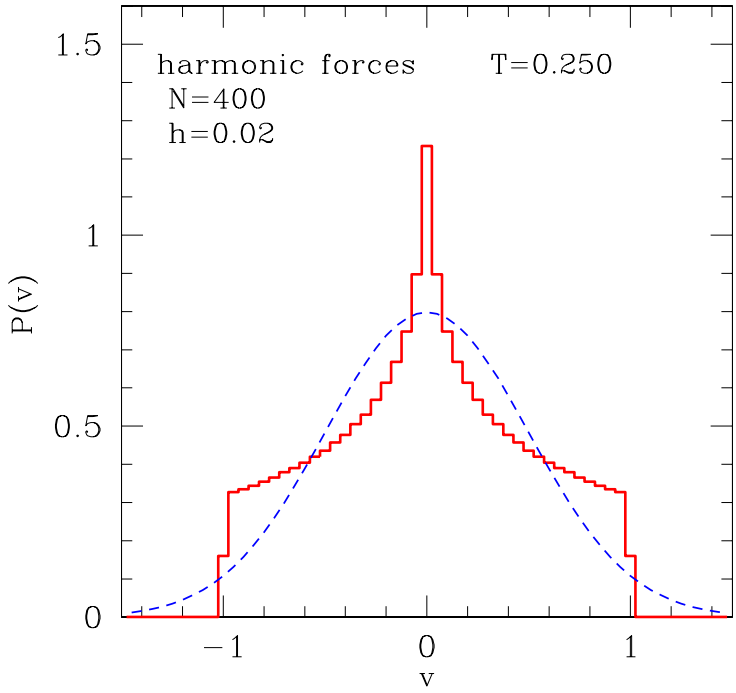
Note that the maximum speed is 1, which arises because the mode with wavevector $2\pi/N$ (into which all the energy is put) is given the maximum speed of 1 (see Eq.(9) in the assignment). Without anharmonicity, this mode oscillates without change, and so the maximum speed stays at 1. By contrast, the Gaussian distribution has a tail which extends beyond 1. Each of the spikes in the MD data corresponds to the maximum (or minimum) velocity of one of the particles. Each particle executes simple harmonic motion with its own maximum velocity equal to the velocity at the initial time.



The details of the spikes depends on the number of particles and also on the phase ϕ in the initial conditions. The above was for $\phi = 0.6$. Below is the corresponding plot for $\phi = 0.78$.



If we increase N to, say, 400, we average over the spikes and get the "envelope" curve:



which is still very far from Gaussian.

SOURCE CODE

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

#define TWOPI   6.283185307179586
double f(double y)                      // Computes the force
{
       return -y - y*y - y*y*y;
//     return -y ;
}

main()
{
    static int   N=20, NH=300;
    double y[N], v[N];
    double h, phi, t_for_equil, t_for_meas, sumv, binsize, v2, u;
    int    i, time, n_for_equil, n_of_meas, n_bet_meas, istep, i_of_meas;
    int    ihist, iv;
    double hist[2*NH + 1];
    void   pos_verlet(int N, double h, double y[], double v[]);

    binsize = 0.05;
    t_for_equil = 20000;
    t_for_meas = 1000000;
    h = 0.01;                       // step size
    phi = 0.6;
    n_for_equil = t_for_equil / h;
    n_of_meas = t_for_meas ;
    n_bet_meas = 1 / h;             // only measure v every unit of time,
```

```
    v2 = 0;                          // not every step
    printf (" N             = %8d\n", N);
    printf (" step size     = %8.4f\n", h);
    printf (" time for equil= %8.0f\n", t_for_equil);
    printf (" time for meas = %8.0f\n", t_for_meas);

    for (i = 0; i < 2*NH + 1; i++) hist[i] = 0; // Initialize histogram
    for (i = 0; i < N; i++)
    {
        v[i] = sin (TWOPI * i / N + phi); // Initialize v's
        y[i] = 0;                    // Initialize y's
    }

    for(istep = 0; istep <  n_for_equil; istep++) // steps for equilibration
    {
        pos_verlet(N, h, y, v);
    }
                                      // steps for measurement, loop over meas.
    for (i_of_meas = 0; i_of_meas <  n_of_meas; i_of_meas++)
    {
        for (istep = 0; istep <  n_bet_meas; istep++) // steps between meas.
        {
            pos_verlet(N, h, y, v);
        }
        for (i = 0; i < N; i++)
        {
            v2 += v[i]*v[i];         // collect data for <v^2>
            iv = NH  + round(v[i] / binsize); // add to histogram
            hist[iv] += 1;
        }
    }

    v2 /= n_of_meas * N;             // Average v^2 is the temperature
    printf ("\n Temperature   = %8.4f \n\n", v2);
                                    // print the histogram
    printf ("      v      P(v)   P(v)(Boltz) \n");
    for (ihist = 0; ihist < 2 * NH + 1; ihist++)
    {
        u = pow((ihist-NH)*binsize, 2);
        if (hist[ihist] != 0 || fabs(u) < 9 * v2) //only print out non-zero entries
        {
            hist[ihist] /= binsize*n_of_meas * N; // normalize the histogram
                                   // print histogram and expected Gaussian
            printf (" %8.2f %8.3f %8.3f \n", binsize*(ihist-NH), hist[ihist],
                    exp(-0.5*u/v2) / (sqrt(2*3.14159*v2)));
        }
    }
}

void pos_verlet(int N, double h, double x[], double v[]) // position Verlet
{
    double f();
    int i;

    for (i = 0; i < N; i++) x[i] += 0.5 * h * v[i]; // half-step in x
//
// Full-step in v; v[0] and v[N-1] treated separately because of periodic bc's
//
```

```
    v[0] +=  h * (f(x[0] - x[1]) - f(x[N-1] - x[0]));
    for (i=1; i<N-1; i++) v[i] += h * (f(x[i]-x[i+1]) - f(x[i-1] - x[i]));
    v[N-1] += h * (f(x[N-1] - x[0]) - f(x[N-2] - x[N-1]));

    for (i = 0; i < N; i++) x[i] += 0.5 * h * v[i]; // half-step in x
}
```

242 students
I took the Fourier transform of the harmonic part of the Hamiltonian and averaged over the measurements. There is a sine-transform and a cosine-transform. For N = 20, some results are shown below

```
#=============================================================================
# N = 20
#
# q = 2 iq PI / N
#
#=============================================================================
#
# Include anharmonic interactions.
#
# Initial energy per particle = 0.25. Total energy = 5.0.
#
  iq     cosFT     sinFT
  0      0.0000
  1      0.2464    0.2460
  2      0.2457    0.2448
  3      0.2452    0.2464
  4      0.2457    0.2430
  5      0.2479    0.2448
  6      0.2459    0.2459
  7      0.2480    0.2463
  8      0.2467    0.2477
  9      0.2454    0.2456
 10      0.2463
#
# Note: there is no energy in the q=0 mode.
# We see that the energy is divided equally between the modes.The
# small differences are probably due to uncertainties from the time average.
# (Note: this calculation of the energy omits the contribution from the
# anharmonic interactions. This is why the sum of the above numbers doesn't
# exactly  equal 5.)
#
#=============================================================================
#
# Purely harmonic interactions
#
# Initial energy per particle = 0.25. Total energy = 5.0.
#
  iq     cosFT     sinFT
  0      0.0000
  1      0.0000    5.0000
  2      0.0000    0.0000
  ...    0.0000    0.0000
 10      0.0000
# i.e. all the energy stays in the mode where it was initially put, as expected.
#
```

The energy is distributed equally among the modes when I include the anharmonic interactions but stays in the longest wavelength mode (where it was put initially) in the harmonic case. These results are just what is expected.