Prof. Mark D Shattuck
Physics 39907 Computational Physics
September 23, 2023

Problem Set 3

Question 1. *2D Volume Fraction:* Create a new MATLAB script based on the MATLAB Molecular Dynamics Simulator `md010.m` found in zip file `mdIntro.zip`. In this new script, add code to measure the 2D volume fraction of the simulation container. The volume fraction is the ratio of the total volume of the particles to the volume of the container. The 2D volume fraction (also called area fraction) ϕ is the ratio of the total area of the particles to the area of the container.

$$\begin{aligned}\phi &= \frac{\sum_{n=1}^{N} A_n}{A_C},\\ &= \frac{\sum_{n=1}^{N} \pi D_n^2}{4L_xL_y},\end{aligned}$$

where A_n is the area of the n-th particle, A_C is the area of the container, D_n is the diameter of the n-th disk (particle), and L_x and L_y are the side lengths of a rectangular container.

(1) Show that for `N=80` mono-disperse (all same-sized) disks of diameter `D=2` in a $9D \times 11D$ rectangular container (`Lx=9*D; Ly=11*D;`) the area fraction is

$$\frac{20\pi}{99} = \frac{80\pi}{396} \simeq 0.6347.$$

(2) Does the area fraction change during the simulation? Why? or Why not?

(3) What is the largest area fraction that we can achieve using the square packing initial condition code from `md010.m`:

```
%% Initial Conditions
[x y]=ndgrid(D/2:D:Lx-D/2,D/2:D:Ly-D/2);
ii=randperm(numel(x),N);
x=x(ii(1:N));
y=y(ii(1:N));
```

(4) Does the maximum realizable area fraction in (3) depend on the box shape or size? Why? or Why not?

(5) For a theoretical infinite system with hexagonal packing, what is the maximum packing fraction? Is there a way to obtain this packing fraction in a finite simulation?

(6) Include the new code (with comments) in the zip file you turn in.

Question 2. *NVE Ensemble:* The molecular dynamics code we developed in class and above is an example of a constant (N)umber, (V)olume, (E)nergy or "NVE ensemble" simulation. Each of these quantities is supposed to be conserved (unchanged) by the simulation. The number of particles N is clearly constant in the simulation. The 2D volume or area A is determine from the values of `Lx` and `Ly`, which are also fixed. So the area of the box is fixed. While `Lx` and `Ly` are fixed the area that the particles can explore actually depends on the overlap of the particle with the wall. We will ignore that for this problem. Newton's Laws conserve energy E so the energy should be fixed. However, we are not solving Newton's Laws exactly. Lets explore how well we the energy is conserved.

(1) Add a new `%% Experimental parameter` to the script from Question 1 to set the initial total energy `E=2;`. Consider the values of the initial potential and kinetic energies separately. The potential energy depends on the overlaps of the particles with each other and the walls. How big are the overlaps at the beginning of the simulation. The kinetic energy is

$$K = \frac{1}{2} M \sum_{n=1}^{N} v_n^2,$$

where v_n^2 is the square of the speed of particle n and

$$v^2 = v_x^2 + v_y^2.$$

For the velocity initial conditions set the values of

```
    vx=randn(1,N);
    vy=randn(1,N);
    % code to determin v0
    v0=???;
    vx=v0*vx;
    vy=v0*vy;
```

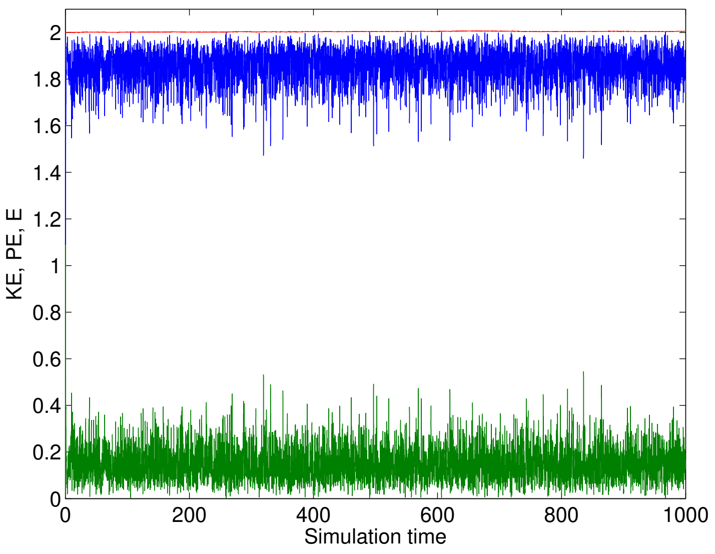
such the `v0` is chosen so that the total initial energy is `E`.

(2) Run the new simulation using

```
%% Experimental Parmeters
N=80;  % number of particles
D=2;   % diameter of particles
K=100; % spring constant for harmonic force law
M=3;   % mass of particles

E=2;   % Total Energy

Lx=9*D;  % size of box
Ly=11*D;

TT=1000; % total simulation time
```

and plot the total potential energy U (particle-particle and particle-wall), the total kinetic energy K, and the total energy $E = U + K$ as a function of simulation time from 0–1000. The result should look like the (left) panel of figure 1. The total time of the simulation should be less than 1 minute. To time the simulation you can use: `tic;md011;toc;`.

(3) Make a second plot of just the total energy. Is the total energy conserved? How well is it conserved?



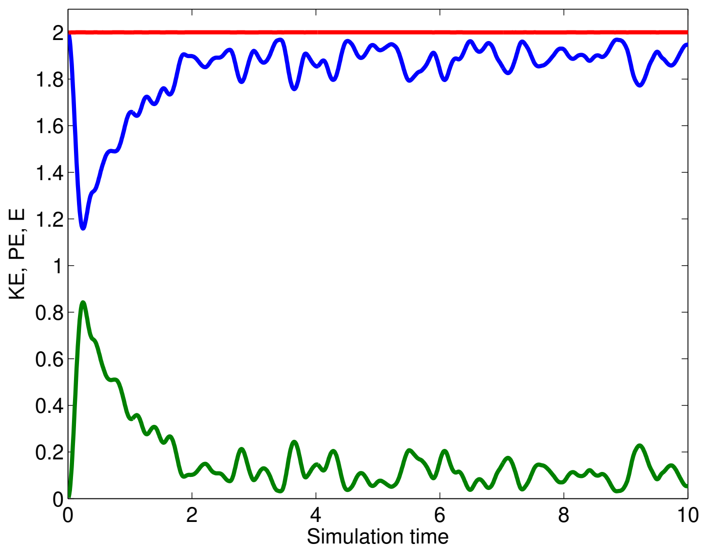


FIGURE 1. Plot of potential (green), kinetic (blue), total (red) energy as a function of simulation time for a $9D \times 11D$ system of 80 disks of diameter $D = 2$. The (Left) panel shows all simulation times, and the (right) panel shows early times.

(4) Run the code 5 times with different values of `dt` from this list `dts=[.05 .02 .01 .003 .001];`. Plot the `std` of the total energy over time as a function of `dt` on a log-log plot. What is the slope of this curve? Explain the slope. Some of these simulation may take a few minutes. A `dt=.01;` is a good compromise between energy conservation and simulation time.

Question 3. *Poly-dispersity:* Poly-disperse is a term used to describe systems with particles that have different sizes. Mono-disperse and Bi-disperse are also common to describe systems with 1 (mono-) or 2 (bi-) sizes. Modify the code to allow each particle to have its own diameter. As a starting point use: `Dn=D*(1+0.1*randn(1,N));` This represents an average diameter of `D` and poly-dispersity of 10% ($0.1D$). You may also want to decrease the simulation time (`TT=100;`) while you are testing the code.

(1) Begin by adding the definition of `Dn` to the `%% Experimental Parameters` section.
(2) Next look through the code for places `D` is used and replace with equivalent versions using `Dn`. For example:
  (a) Calculation of area fraction: `phi=pi*N*D^2/4/Lx/Ly;`. `N*D^2` is replace by `sum(Dn.^2)`. Show that this reduces to `N*D`, if `Dn=D*ones(1,N)`.
  (b) Initial Conditions: `[x y]=ndgrid(D/2:D:Lx-D/2,D/2:D:Ly-D/2);` Here there are a few options, but for this code use the maximum of `Dn`, `Dmax`, in place of `D`. You may want to put the calculation of `Dmax` in the `%% Experimental Parameters` section of the code so that it is available in most of the code. Make sure you calculate `Dmax` *without* a `for` loop. With this solution the code will not accommodate as many particles. Approximately 55 particles will fit in a $9D \times 11D$ container.
  (c) Plotting: `...'Position',[x(np)-.5*D y(np)-.5*D D D]...` you will need to replace `D` with the value of `Dn` for particle `np`.
  (d) Force Law: Here `D` must be replaced by the sum of the radii of the interacting pair `Dnm`. The same is true for the particle-particle potential energy `Ep`.
  (e) Wall Forces: For each wall force:

```
ii=x<D/2;      % list of overlaping particles
dw=x(ii)-D/2; % Left wall overlap size
```

  `D` appears in making the list of overlaps and in calculating the size of the overlap.
  (f) Search the code for `D` to make sure you have found all uses of `D`.
(3) When the code is working it should still conserve energy. This is a good way to check that you have done it correctly.
(4) It would be unusual to have particles with different diameters but the same mass. Change the code to allow different masses for each particle, as we did in class. A good choice for the masses is to use a constant mass per unit area (areal density `rho=1;`). Then the `M=rho*pi*Dn.^2/4`. After the change make sure that energy is conserved, and the value of the total energy is still `E`.
(5) Include this code in the zip file you submit.

Question 4. Ideal Gas Law: Use the code from Question 3 to check the ideal gas law. The ideal gas law is usually expressed as:

$$PV = NRT.$$

The pressure times the volume is equal to the number of particles times a constant times the temperature. From the simulation we know the number of particles `N` and the 2D volume or area `Lx*Ly`. The remaining terms need some discussion:

R: The ideal gas constant R is really a units conversion for the temperature as measured in Kelvin to the temperature as measured in Joules. So we can eliminate it from the equation if we measure temperature in the right units.

T: In the kinetic theory derivation of this equation the temperature is average kinetic energy of the system per particle. Therefore,

$$T = \left\langle \frac{1}{N} \sum_{n=1}^{N} K_n \right\rangle_t ,$$

where

$$K_n = \frac{1}{2} M_n v_n^2$$

is the kinetic energy of particle n, and v_n^2 is the square of the speed of particle n and

$$v^2 = v_x^2 + v_y^2.$$

The average $\langle ... \rangle_t$ is over time. In the code, the kinetic energy at each time step is saved as `Ek`. To get the average kinetic energy, we need to find the mean of `Ek`. However, we want the system to be in steady-state when we make our measurements. As shown in the (right) panel of figure 1 we start the simulation in an unusual state where all of the energy is kinetic. It takes at least 5 simulation time units before K and U reach steady-states. To account for this relaxation, run the simulation for 1000 simulation time units (`TT=1000;`) but do not use the first 10% of the data to calculate the average kinetic energy. MATLAB has a simple way of implementing this (`K=mean(Ek(fix(.1*end):end));`). `end` is a shorthand for the length of the vector, and `fix` returns the integer part of a number.

P: To measure the pressure P, we need to find the average force on the walls per unit length. This is the 2D analog of force per unit area. The current code does not save the force on walls, but does save the potential energy. For example, for the left wall: `Ewp(nt,1)=K*sum(dw.^2)/2; %PE`. By analogy, update the code from Question 3 to measure the force on each wall over time in a variable like this: `Fws=zeros(Nt,4);`.

(1) As a check of the updated code with force measurements, run the code for these conditions:

```
%% Experimental Parmeters
N=80;  % number of particles
D=2;   % mean diameter of particles
pd=0;  % particle diameter poly-dispersity
rho=1; % areal density

Dn=D*(1+pd*randn(1,N));  % list of particle diameters
Dmax=max(Dn);            % maximum particle diameter

```

```
M=rho*pi*Dn.^2/4;    % mass of particles

K=1000; % spring constant for harmonic force law

Lx=3*9*D;  % size of box
Ly=3*11*D;

E=10;  % Total Energy

TT=1000; % total simulation time
```

To simplify the theory the disks are mono-disperse `pd=0;`, and `N=80;` to increase averaging. `K=1000;` so the particle will overlap less, and `E=10;` to increase the number of collisions. The ideal gas law only applies to dilute systems, so this box has $9 = 3 \times 3$ times the original area, but is the same shape as our previous examples. For these conditions the force per unit length *on* each of the four walls *from* the particles is approximately `-0.0034 -0.0030 0.0032 0.0031` for the left, top, right, bottom. Pressure is tensor so we need to define a sign convention. One common convention is that pressure is positive when the particles are pushing the walls outward.

(a) For the force per unit length above, explain why all of the pressures are positive.
(b) Why are the pressures different on each wall?
(c) Are the numbers the same each time you run to code? Why? Why not?

To get the pressure average all 4 wall pressures (`0.0032`). Now we can check the ideal gas law. A nice way to express The law for the case of $R = 1$ as explain above is to look at the quantity

$$Z = \frac{PA}{NT}$$

which is 1 if the law is true. For the example above, `P=0.032`, `T=0.125`, `A=Lx*Ly=3564`, `N=80`, and `P*A/N/T=1.17`.

(2) Re-run the code above 5 times to estimate the error in Z. Report the average value of Z and the standard deviation for the 5 runs.

(3) For the next step we want to be able to run the code for a particular area fraction `phi`. Currently `phi` is an output of the code. To make it an input we need to calculate `Lx` and `Ly` from `phi`. However `phi` cannot determine both lengths, so we can fix the aspect ratio `G=Lx/Ly` to `9/11`. Using the equation above for `phi` and the definition of `G` show that `Lx=sqrt(pi*G*sum(Dn.^2)/phi)/2;` and `Ly=Lx/G;` will make a system with area fraction `phi`.

(4) Measure Z for 20 values of the area fraction `phi` from 0.02 to 0.7, holding `N`, `D`, `E` fixed. To make the `phi` list in MATLAB use `phi_list=.02+(0:19)/19*(.7-.02);`. One way to do this is to do it by hand.

(a) Change `phi` in the simulation.
(b) Run the simulation.
(c) Determine and save Z.
(d) Repeat for all `phi` in the list.

However it may be useful to make a script to do this. Here is an example,

```
phi_list=.02+(0:19)/19*(.7-.02);

Nphi=length(phi_list);

Zs=zeros(1,Nphi);
```

```

for nphi=1:Nphi        % loop over all area fractions
  phi=phi_list(nphi); % set phi to np-th value

  disp(nphi);          % user feedback
  tic; md011; toc;     % run simulation

  Zs(nphi)=calcZ(Fws,Ek,Lx,Ly,N); % calculate and store Z

  % plot results as they are calculated
  figure(2);
  plot(phi_list,Zs,'.');
  drawnow;
  figure(1);
end
```

The script makes a list of `phi`. Then in a `for` loop sets `phi` from the list. You should replace the function `calcZ` with your own code to calculate `Z`. The key for this to work is that the simulation `md011` does not set `phi`, so comment out the line that sets `phi`. Also, you can not use any variables in the script that are also in the simulation code.

(5) Plot `Z` vs. `phi`. It should approach 1 as the area fraction goes to zero. However, as ϕ increases the area that the particles can explore get smaller. A simple correction is to remove the area taken up by the particles themselves. Show that this leads to a correction of the form $Z = 1/(1-2\phi)$. (Hint: Draw a 4×4 box with a disk of diameter 1 in the center. Now shade the area where the center of another particle of the same diameter could be. What is the area of the excluded region? This excluded area is shared by the pair of particles so the total exclude area for each is 1/2 of this area.) This correction is due to Van der Waals. One problem with Van der Waals correction is that it diverges at $\phi = 1/2$. One solution is to just use the linear part $Z = 1/(1-2\phi) \simeq 1 + 2\phi$, since the excluded volume is only correct when the density is low. Another common ad hoc correction is:

$$\begin{aligned}
Z &= \frac{1}{1-2\phi} \\
&= 1 + \frac{1}{1-2\phi} - 1 \\
&= 1 + \frac{1-1+2\phi}{1-2\phi} \\
&= 1 + \frac{2\phi}{1-2\phi} \\
&\simeq 1 + \frac{2\phi}{1-2\phi+\phi^2} = 1 + \frac{2\phi}{(1-\phi)^2}
\end{aligned}$$

A slightly less ad hoc method due to Carnahan and Starling gives this approximation:

$$Z = 1 + \frac{2\phi - 7/8\phi^2}{(1-\phi)^2}$$

Add all of these approximations:

$$Z = 1 \qquad \text{Ideal}$$

$$Z = \frac{1}{1-2\phi} \qquad \text{first-order exculed area (Van der Waals)}$$

$$Z = 1 + 2\phi \qquad \text{linear Van der Waals}$$

$$Z = 1 + \frac{2\phi}{(1-\phi)^2} \qquad \text{ad-hoc Van der Waals}$$

$$Z = 1 + \frac{2\phi - 7/8\phi^2}{(1-\phi)^2} \qquad \text{Carnahan and Starling}$$

to your measured `Z` plot. For the plot limit the x-axis to 0–0.75 and the y-axis to 0–15. In MATLAB `axis([0 .75 0 15])`. For $Z = 1/(1-2\phi)$ only plot up to $\phi = 1/2$.