PYU33A/P15 Statistical Thermodynamics

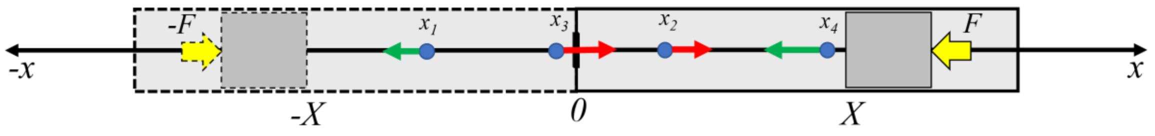
CA2 Programming Project

Instructions:

- This must be your *own work*. Some collaboration is expected, but your submitted project must be done on your own.
- Justify and explain your assumptions and steps with English sentences
    - Just writing down math will not get full marks, even if correct!
- It is strongly recommended that this project be completed using the **Python** language as you have used in your other programming courses in JF, SF and JS.
    - The **Spyder** programming environment is recommended for this to easily edit and execute your source code. If you need to install Python and Spyder on your computer, this is most easily done using the **Anaconda** package manager (easily found via Google.)
    - For the more adventurous, you may attempt to use **Julia** which will be similar (but better ;) I do **not** recommend this unless you are a confident programmer already. In this case, you could use install **Jupyterlab** as a programming environment.
- The assignment is to be submitted in two files:
    a. All analytical work, code, printed output and graphical output of your computer programs should be submitted withing a single PDF. *NOTE: Your mark is based on the content of this document, so include **all** your code and output here.* Treat this file like a report which must include discussion of your results. Simply pasting graphs will not get full marks.
        i. You can use LaTeX or Microsoft Word, and then include graphs and printouts from your computer
    b. Your working mini-exercises and main program should also be submitted as a single source code file (eg. a `.py` Python file, not a Word or PDF file.)
        i. When run, this `.py` file should first print out your name and student ID, and then compute and generate **all** the textual output and plots used in your PDF.

Analysis of an ideal gas in one dimension from "first principles"

In this simple **molecular dynamics** problem, you will consider a system comprising N *classical* point particles of mass m allowed to move only back and forth in one dimension along a half-line at positions $x_i \geq 0,\ i=1, 2, ..\ N$. These molecules are contained between a wall at the $x=0$ origin and a piston of mass M at position $X > 0$. The piston is acted on by a constant force F pointed towards the origin. The particles move under Newtonian dynamics where their only interaction is to simply bounce elastically off the rigid inner wall or scatter off the piston in an elastic collision (they simply pass through each other without interacting otherwise!)



HINT: A trick to ease the calculation for this problem is to extend the particle path to negative values of x and imagine that there is an identical piston at position -X like in the figure above. This simplifies treatment of the elastic bounce at the rigid inner wall, and means you only have to consider collisions with the piston at +X for particles moving to the right and the piston at -X for particles moving to the left.

1. Consider the following analytical aspects of the problem and write answers in your PDF file using Word or LaTeX or scanning neat handwriting. For a **microstate** of the system at some time t_0 given by the collection of position and velocity variables (small letters are particles, capitals are the piston)

$$(x_1^0, v_1^0, x_2^0, v_2^0, \cdots x_N^0, v_N^0, X^0, V^0)$$

   a. What is the internal energy *U* of the system in terms of the above microstate variables? [1 mark]
   b. What is the enthalpy *H* of the system? [1 mark]

   c. What are the equations of motion $x_i(t)$ and $v_i(t)$ of the *i*th particle for $t \geq t_0$? (Hint: these are very simple..) [1 mark]
   d. What are the equations of motion $X(t)$ and $V(t)$ of the piston for $t \geq t_0$? [2 marks]
   e. For a particle with (x,v) and piston with (X,V) at time t_0, show that the **waiting time** τ to a collision of the particle with the piston is given by

$$\tau = \frac{M}{F}\left(V - v + \sqrt{(V-v)^2 - 2\frac{F}{M}(x-X)}\right) \quad v>0 \quad \text{(right moving particle)}$$

$$\tau = \frac{M}{F}\left(V + v + \sqrt{(V+v)^2 + 2\frac{F}{M}(x+X)}\right) \quad v<0 \quad \text{(left moving particle)}$$

   NB: x and X are positive in these expressions. [3 marks]

   f. Assuming purely elastic collisions preserving kinetic energy and momentum, show that the velocities *W*, *w* of the piston and a particle immediately after the collision between them are given by

$$W = \frac{\pm 2mv + (M-m)V}{m+M}$$

$$w = \frac{\pm 2MV + (m-M)v}{m+M}$$

   where *V* is the velocity of the piston just before the collision, and *v* the velocity of the particle just before the collision. The + of the ± is for a right moving particle ie. $v > 0$, and the – for a left moving particle, $v < 0$.

   [2 marks]

2. The following Python (or Julia) programming mini-exercises may help you with writing the main program. Compose code to do the following (*include your code and a* ***screen capture*** *of its output directly in your PDF.*)
   a. Create a 1000x2 array `z` initialized to have all zero-valued elements using a one-liner Python statement. [1 mark]
   b. Define a function with input arguments consisting of an array `z` of floating point numbers along with two additional floating point numbers `a` and `b`. The function should return the sum of the elements of the array to the power of `a` and

`b` as two numbers separated by commas. Call the function with a 10 element array of random numbers between 0 and 1 and pass values 2 and 3 for `a` and `b`. Place the returned number pair into variables `x` and `y`. `[1 mark]`

c. Create a real-number ("floating point") array of 500 random valued elements drawn from a Gaussian distribution. Plot a histogram of these values showing at least ±3 sigma (standard deviations) with 20 bins. [1 mark]

d. Create an array of 20 elements chosen randomly to have the value +10 or -10 (NumPy `choice()` function). Create and print out this array 3 times. [1 mark]

e. Create an empty array (eg. `z=[]`), and then use a 100 iteration `while` loop inside a 50 iteration `for` loop to create and append 5000 random numbers uniformly distributed between -50 and +50 using `z.append(<number>)`. After the loop, plot a histogram of `z` with 30 bins. [1 mark]

3. The main programming exercise. (*Include your code and its output in your PDF.*)

Your algorithm to evolve the simple 1D molecular dynamics system might look like this:

I. **Initialize** the system by setting the initial piston position X_0 to be at twice (ie 2x) the equilibrium position as given by the 1D ideal gas law for a temperature T_0. (*Think, what does this do to the initial internal energy of the system?*) Then distribute the N initial particle positions x_i^0 randomly to be *between* -X_0 and X_0.

   i. You can use the equipartition theorem to set a reasonable scale of the initial velocity of the punch and particles: Assume $K = \frac{1}{2}k_B T$ goes into the kinetic energy of each particle and the punch. For part 3(a), you can set the v_i^0 to be a random Gaussian value with a standard deviation of particle velocity as calculated from K, and same for the initial punch velocity V_0. Note for part 3(c), you should set the v_i^0 to be a random value of +v0 or -v0 as set by the K calculation to see the evolution to the Maxwell distribution.

   ii. Set an overall time measure variable t to 0.

   iii. To hold the current microstate variables of the particles, it might be convenient to define an Nx2 array with column 1 holding particle positions and column 2 holding particle velocities. Or you could make two separate Nx1 arrays. You could extend the array to length N+1 to hold the piston position and velocity (ie. representing the *full* microstate), or just keep those as separate variables.

II. Evaluate the waiting times τ_i to a collision of each particle i with the piston using answer 1e above and identify the index i_0 of the first collision with the shortest waiting time. This smallest waiting time will be the time of the next step to evolve the system.

   - The waiting time calculation might be conveniently placed in a **function** with inputs x, v, X, V, and F/M, and returning τ, to be called by a loop over all the particles.

III. Displace each particle i according to the equations of motion from answer 1c, place the piston at the position of particle i_0 (ie. the one you found it is colliding with), and update the velocity V of the piston from answer 1d.

IV. Use the answer 1f to calculate new values of velocity of the particle and piston after the collision.
    - The new velocities calculation may also be conveniently placed inside a function that returns two values at once (eg. in Python you can write `return w, W`)

V. Increase the time measure by τ_{i_0}. Store the values of τ_{i_0}, *X*, and *V* computed at this time step in arrays that will represent a *time series* (ie. sequence of time-varying physical values vs. time) of the piston motion to be plotted when the main loop is done.
    - In Python, you can write `x=[];` then use `x.append(y)` to store values of a variable `y` to the end of array `x`, say as `y` is computed again and again inside of a loop.

VI. Loop back to step II, repeating enough times to realize at least 20 physical oscillations of the punch position as it settles down about an equilibrium value. This value should correspond to that predicted by the 1D gas law. To do this, the number of repeats of this outer loop should be something like 20 times the number particles N you have.

The problems to be computed are:

a. Compose a program to calculate and plot a time series of the motion of the piston as it performs a damped oscillation towards equilibrium (eg position X(t) vs t.) Each time step should be recorded for the next collision of the piston with a particle. Run the program for enough time steps to get 20 or more oscillations for $N=1000$ particles. You should use the following initial values (unitless, with $k_B=1$): particle mass $m=1$, piston mass $M=100$, initial temperature $T_0=1$, force on piston $F=10$ (pointing towards origin). The initial piston position X_0 should be twice the value predicted by the 1D ideal gas law for $T_0=1$ (see further comments above on initialization of the particle positions and velocities.) Calculate the expected piston equilibrium position for these initial conditions and show this as a dotted horizontal line on your plot. Comment on all features of your results. [15 marks]

b. Show that your system agrees with the equation of state of the 1D ideal gas by running your program (ie. (a) above) and calculating the average piston position $\bar{X}$ over several final oscillations. (Hint: you should use a weighted average since the time steps are of different values, see the Python function `average(z,weights=w)` where `x` is an array of numbers and `w` is a similar size array of numbers representing the weighting for each.) Plot $\bar{X}$ for for 7 values of force *F* over 3 decades (say spread 0.1, 0.3, 1, 3, … to 100) and compare directly by plotting the 1D ideal gas theory curve on top. Use base 10 log-log axes on your plot. To make things easier, you can set the initial piston position X_0 to start at the predicted ideal gas law value already. [8 marks]

c. Show that an initial bi-polar distribution of the particles' velocity tends towards the Maxwell distribution by storing all the velocities and plotting a sequence of histograms at a selection of time steps (say at least 7) as equilibrium is approached. Here you should change the initial particle velocities to be a random two-valued distribution (see comments above on initialization.) Be sure you record at fine enough time intervals to really see the evolution from the bi-polar to Maxwell distribution. Fit an appropriate theory curve and report the mean and standard deviation of the fit. Comment on why the system thermalizes without interactions between the particles.
[12 marks]

d. Bonus activities
    i. Check that the **enthalpy *H*** of this *constant force* system is conserved at all times as the system approaches equilibrium by plotting *H* vs. *t*

ii. Redo parts 3(a),(b) and (c) using SI units. Comment on what the physical size of system must be for the non-interacting gas molecule interpretation to be true at standard temperature and pressure.
iii. If you run the system much longer, does it settle down asymptotically to the ideal gas expected value, or does it keep fluctuating? What happens if you increase and decrease the value of N by a factor of 10?
iv. How could you add some weak interaction between the particles? Try this with a smaller number of particles to see how the system thermalizes.

[5 extra marks, prof's discretion, max is still 50]