University of Notre Dame

Statistical Analysis of Classical Ideal Gases using Python

Isaiah Mumaw

March 8, 2021

1. Introduction

The classical ideal gas is an approximate model of a real gas in which interactions between particles are ignored. It is usually valid within the low pressure/high temperature limit, where the size of particles is negligible compared to the space between them, and the potential energy due to intermolecular forces is negligible compared to the translational kinetic energy. This provides a simple relationship between the pressure, temperature, and volume of the gas [1].

Using statistical mechanics, one can easily describe the behavior of a large number of ideal gas particles. In a three-dimensional box, the probability distribution of the component velocities of ideal gas particles is described using the Maxwell-Boltzmann distribution:

$$f_{3D}(v)\,d^3v = \left(\frac{m}{2\pi k_B T}\right)^{3/2} \exp\left\{-\frac{mv^2}{2k_B T}\right\} d^3v \tag{1.1}$$

where k_B is the Boltzmann constant, m is the mass, and T is the temperature.

While this may be useful in some niche cases, a more useful distribution is that of the speeds, as the overall kinetic energy is solely based on the magnitude of the velocity. This distribution can be found by integrating Eq. 1.1 over a spherical shell of radius v:

$$\iint_S f_{3D}(v)\,dv\,dS = f_{3D}(v)\,dv = \left(\frac{m}{2\pi k_B T}\right)^{3/2} 4\pi v^2 \exp\left\{-\frac{mv^2}{2k_B T}\right\} dv \tag{1.2}$$

The energy distribution can then be derived from this through a change of variables:

$$f_{3D}(E) = 2\sqrt{\frac{E}{\pi}}\left(\frac{1}{k_B T}\right)^{3/2} \exp\left\{\frac{-E}{k_B T}\right\} \tag{1.3}$$

While this distribution is entirely valid, there is a much simpler computational option available. After randomly generating a large array of velocities using Eq. 1.2, these values can be plugged in to the kinetic energy equation. This gives the same distribution as before, but is much easier to work with.

$$E = \frac{1}{2}mv^2 \tag{1.4}$$

As a test case for this project, we will use the far simpler two-dimensional ideal gas. It is preferable in this case to use the energy distribution as the starting point, since it is much more straightforward:

$$f_{2D}(E) = A\exp\left\{-\frac{E}{k_B T}\right\} \tag{1.5}$$

The velocities, then, are given by slightly reordering Eq. 1.4 (not shown) and applying the same process as with the three-dimensional gas.

If multiple particles are drawn from an ideal gas, the overall distribution is given as the average of the distributions for each particle. As the number of particles is increased, the resulting distribution tends towards a Gaussian. This is known as the Central Limit Theorem (CLT), and is true for any set of distributions.

The resulting Gaussian for N particles with the same distribution is given as:

$$P(Y = y) = \frac{1}{\sigma\sqrt{2\pi N}} \exp\left\{-\frac{(y - N\mu)^2}{2N\sigma^2}\right\} \tag{1.6}$$

where $Y = \sum_i^N E_i$, and σ, μ are the standard deviation and mean of the original distribution, divided by N. E_i is the energy of a single particle i.

2. Coding Procedure

The distributions in this project were programmed using scipy.stats. For the two-dimensional energy distribution, we used scipy.stats.expon.rvs(), and then wrote a separate function for the speed which relied on the results of this function. For the three-dimensional speed distribution, we used scipy.stats.maxwell.rvs(), and used a similar process as before to get the energies. The .rvs method allows us to get randomly generated numbers based on a predetermined distribution.

The plotting function for the distributions takes a sampling of values and applies the pyplot histogram function to the results. Results are normalized by default using the density parameter.

The energy analysis function generates an array of energies from the distributions at each temperature index, adding it to an array and then plotting the results using pyplot's plot function. It is somewhat slower than the previous function as it relies on a "for" loop and has to repeatedly generate new energy arrays at each loop.

The speed analysis function gave us the most trouble. The generating process is identical to the energy analysis (and equally slow). Average and RMS speeds are calculated using basic numpy functions. Most likely speed is found by locating the tallest bin in a histogram and saving the associated velocity. However, this histogram method produces very noisy data, even after finding the optimal input parameters, because the array generating process is still random and thus has some unpredictable variation for finite arrays (see Fig. 2.1).

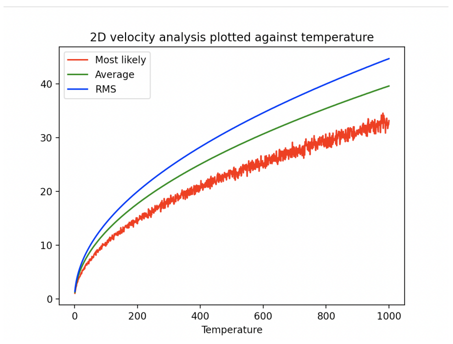


Figure 2.1: Noisy analysis plot, calculated with the test case of $k_B = m = 1$.

In order to remedy this, we applied a Savitzky-Golay filter to the final array of most likely velocities. We used a window of 7 with a polynomial of degree 1, which managed to smooth the results without sacrificing too much information at lower temperatures. The results can be seen in the next section.

For drawing N particles, we generated an array of energies, and then an array of random indices with N rows. Energy values are selected using these indices to create an array of equal dimension. These selections are then averaged along the columns and plotted. The Gaussian is plotted based on the initial energy array for a single particle.

An alternative way of doing this would be to create a loop and generate a new array of energies at each step in the loop. However, our method is much faster because it generates far fewer points and executes the whole process at the same time, rather than in incremental steps.

3. Plots

We first plotted the energy and velocity distributions by generating 1×10^6 weighted random numbers from Eqs 1.2, 1.4, 1.5, and the reordering of kinetic energy for two dimensions. The results are shown in Figs. 3.1 and 3.2. In these and all following plots, $k_B = 1.380649 \times 10^{-23}\,\mathrm{J\,K^{-1}}$ and $T = 300$ K, except where otherwise noted. The mass is set to that of a Helium atom ($6.6464731 \times 10^{-27}$ kg), since Helium most closely matches the behavior of an ideal gas.

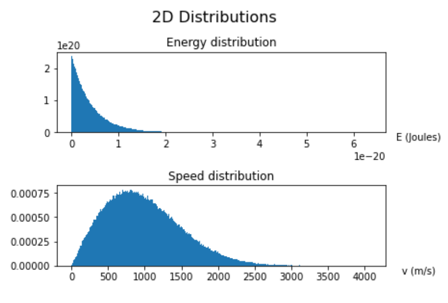


Figure 3.1

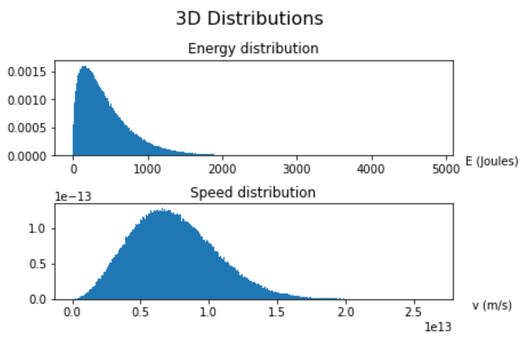


Figure 3.2

The average, variance, and standard deviations of the energy distributions (in Joules) are shown in Figs. 3.3 and 3.4. Results are plotted with respect to temperature, from $T = 1$ K to $T = 1000$ K. Each temperature step generated 1×10^4 energy values.

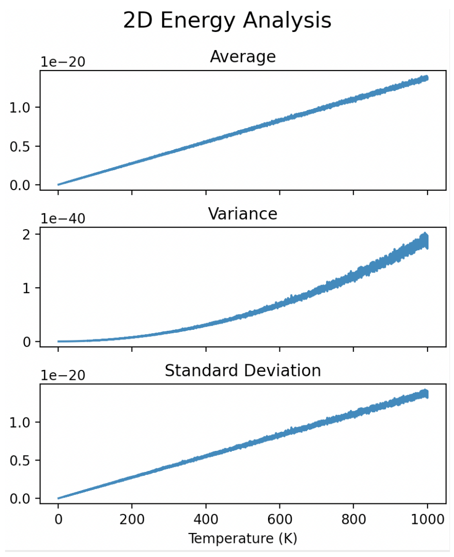


Figure 3.3

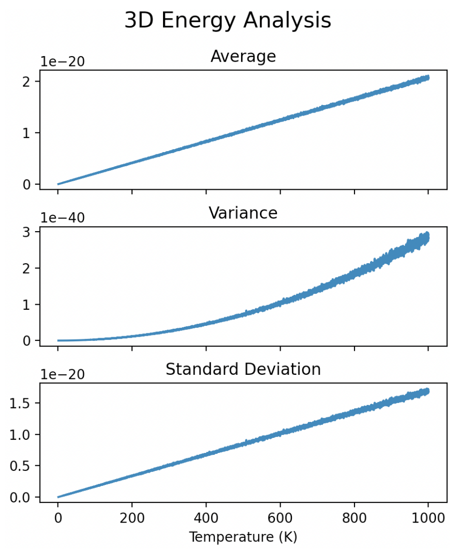


Figure 3.4

We also found the most likely, average, and RMS velocities with respect to temperature, again from $T = 1$ K to $T = 1000$ K. Each temperature step generated 1×10^6 speed values. The results are shown in Figs. 3.5 and 3.6.

Finally, for an N-particle system, we generated plots for 2, 4, 8, and 16 particles. The plots for 2 and 16 particles are shown in Figs 3.7, 3.8, 3.9, and 3.10; the rest can be found in Appendix A (including some additional plots for 30 and 100 particles). Each plot was generated using 1×10^6 energy values for the first particle.

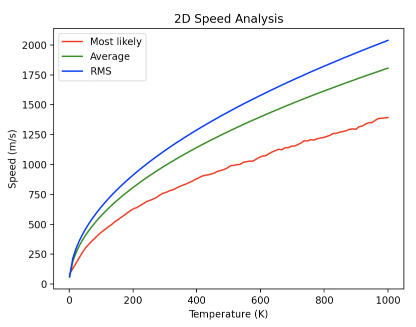


Figure 3.5

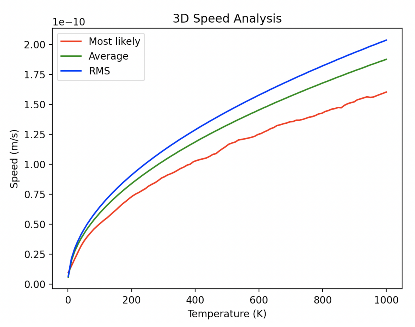


Figure 3.6

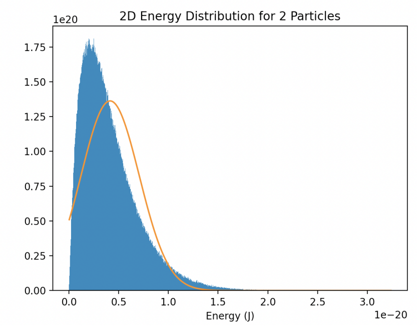


Figure 3.7

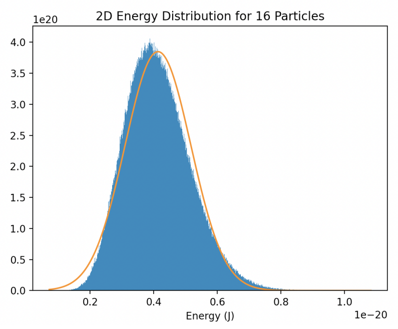


Figure 3.8

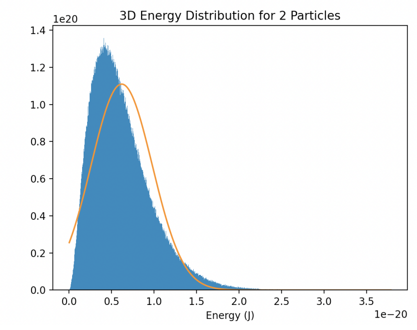


Figure 3.9

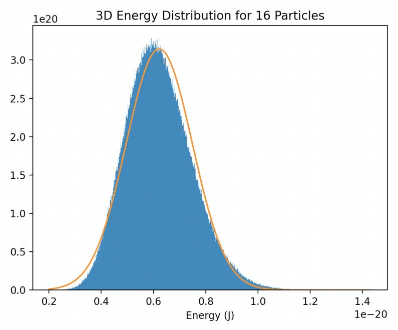


Figure 3.10

4. Observations and Analysis

The distributions given in Figs. 3.1 and 3.2 match expectations, with the two-dimensional gas energy decaying exponentially, while the three-dimensional gas energy peaks at a nonzero value due to the square root term in its distribution equation (Eq. 1.3). The corresponding speed distributions are also as expected, with nonzero speeds dominating, and the three-dimensional gas having much greater speed values in general.

The energy analyses given in Figs. 3.1 and 3.2 also match known results. According to the Kinetic Theory of Gases, the average energy per molecule is as follows [3]:

$$\overline{K_{2D}} = \frac{1}{2}m\overline{v}^2 = k_B T \tag{4.1}$$

$$\overline{K_{3D}} = \frac{1}{2}m\overline{v}^2 = \frac{3}{2}k_B T \tag{4.2}$$

This is exactly the result we see in our plots, with the average energy increasing linearly in both cases, and the average energy for a three-dimensional gas increasing at a rate $\frac{3}{2}$ times faster than a two-dimensional gas.

Based on these equations, the standard deviation should also be linear with temperature, and this is shown exactly in the plots. As temperature increases, the energy probability curve flattens. And since variance is the square of standard deviation, we also get the expected result here.

From the kinetic theory of gases, we can also verify the behavior in Figs. 3.5 and 3.6. By rearranging Eqs. 4.1 and 4.2, the average velocity should be proportional to the square root of the temperatures, which is the exact behavior shown in the graph. Known behavior of gases predicts that the most likely speed and RMS speeds should be related to the mean by a constant, with the most likely speed being slightly lesser and the RMS speed being slightly greater [2].

Finally, when drawing a group of N particles from the distribution, CLT dictates that the distribution should get narrower and taller around the mean with each additional particle introduced, gradually approaching a Gaussian curve. This is exactly the phenomenon observed in Figs 3.7 through 3.10, with the two-dimensional particles having a net average energy of 4.141×10^{-21} J and the three-dimensional particles having an average energy of 6.206×10^{-21} J.

5. Discussion

The results we obtained confirmed previous analytical analyses of both two- and three-dimensional ideal gases, including their behaviors under varying temperatures. The results also reaffirmed the validity of CLT, which is an important statistical theorem with far-reaching applications beyond statistical mechanics.

While every effort was made to optimize this code, the energy and speed analyses were initially plagued by the limitations of Python. "for" loops are notoriously slow, but unfortunately could not be avoided in this case. While scipy and numpy are vectorized, the built-in distribution functions could not handle an array for the scale parameter, and had to be recalculated at each step. After some trial and error, we did find it was possible to speed things up by lowering the array size, however this was at the cost of some accuracy.

This code could easily be adapted to similar physical problems dealing with energy and velocity distributions, such as in more exotic or theoretical states of matter or in more trivial non-ideal gases. Using methods such as the Metropolis-Hastings algorithm, one can also adapt this code to work for distributions outside those that have been previously defined, which opens up nearly infinite options for the application of these methods.

References

[1] Schroeder, Daniel V. (2000). *An Introduction to Thermal Physics.* Addison Wesley Longman.

[2] Wikipedia (2021, January 19). *Maxwell–Boltzmann distribution.* https://en.wikipedia.org/wiki/Maxwell–Boltzmann_distribution

[3] Wikipedia (2021, February 21). *Kinetic Theory of Gases.* https://en.wikipedia.org/wiki/Kinetic_theory_of_gases

Appendices

A. All generated plots for N-particle systems

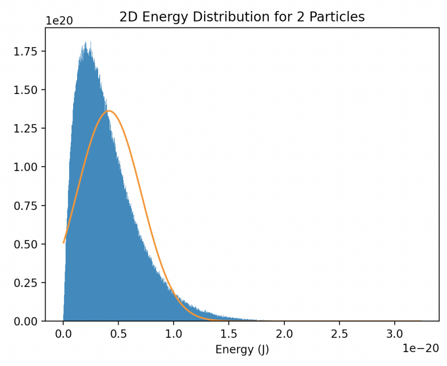


Figure A.1

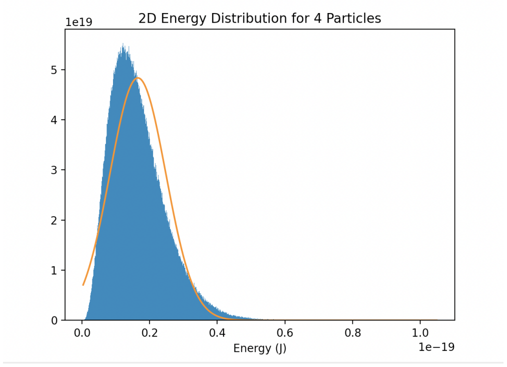


Figure A.2

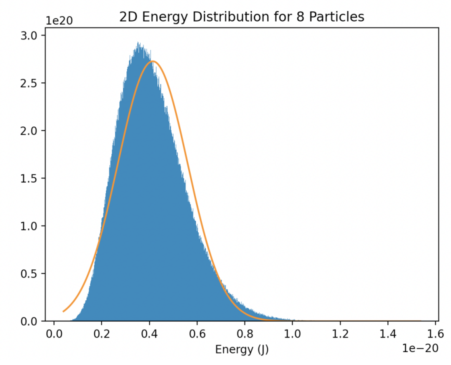


Figure A.3

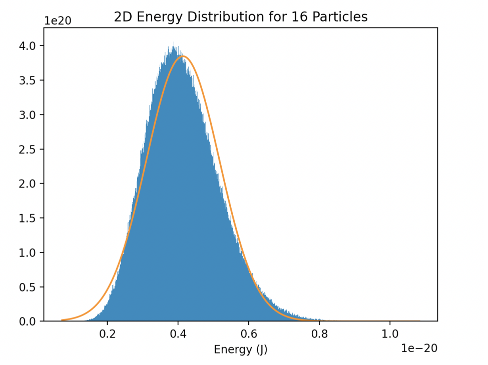


Figure A.4

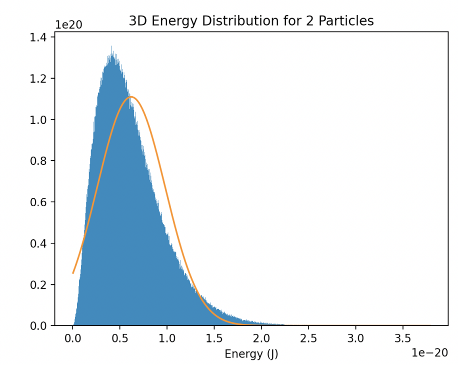


Figure A.5

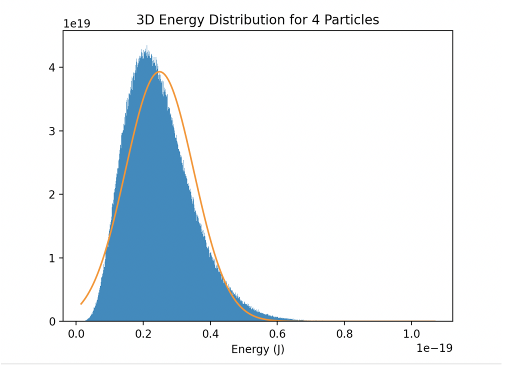


Figure A.6

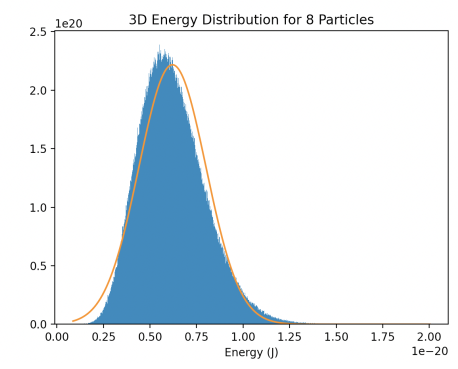


Figure A.7

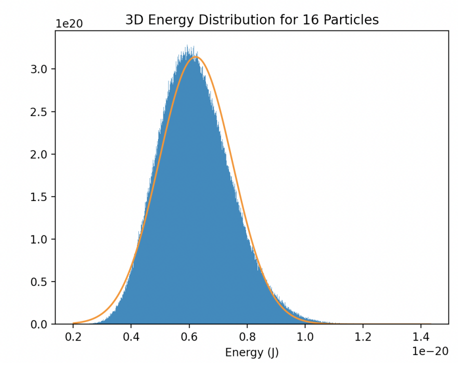
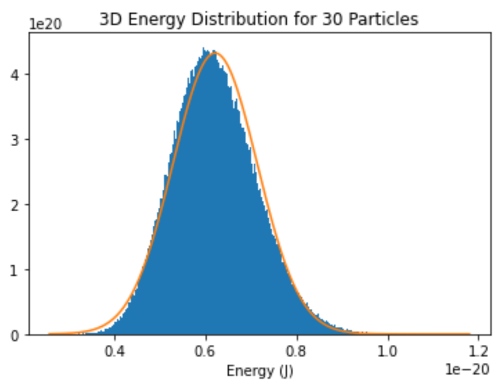


Figure A.8



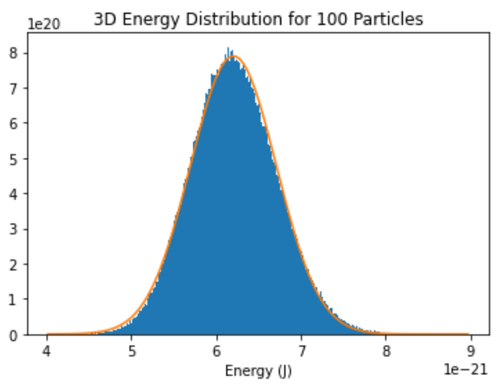


Figure A.9

The central limit theorem is typically most relevant when N>30, so I have included plots for 30 and 100 particles. Note the very close fit for 100 particles.