Generation and Analysis of Gas Particle Velocities from the Maxwell-Boltzmann Distribution

Neema Rafizadeh[*]
Department of Physics and Astronomy,
University of Kansas, Lawrence, KS, 66045, USA

(Dated: April 10, 2023)

This experiment tested the ability for a group of python scripts to generate large sets of data according to a Boltzmann distribution with a given parameter T, and then determine from the samples what the parameters were for each set. The trend found for this process showed a positive correlation between the true value and the calculated value, with some uncertainty.

I. INTRODUCTION

Many physical systems studied in modern physics research follow the Maxwell-Boltzmann (M-B) distribution. From the study of electron diffusion in the semimetal graphene to the modeling of photon interactions between a laser and a solid in photonics, the M-B distribution has proven to have great utility in explaining observed physical phenomena. The distribution has more fundamental roots in its derivation from the first principles of mass, temperature, and velocity of gas particles. However, the question becomes: how well can a python script predict the temperature parameter of one of these curves?

II. HYPOTHESES OF DIFFERENT TEMPERATURE GASES

The M-B distribution is derived from basic physics principles based on some assumptions, namely that the gas particles are non-interacting, non-relativstic, and in thermal equilibrium. If this is the case, then the probability distribution that you find a particle with a particular velocity is given by,

$$f(m, v, T) = \left(\frac{m}{2\pi kT}\right)^{1/2} 4\pi v^2 e^{-\frac{mv^2}{2kT}},$$

where m represents the molecular mass of the particles and T represents the equilibrium temperature of the gas. Two examples of this probability distribution are plotted in Fig. 1 with temperatures 100 K and 1000 K.

Note this is a probability *density*, not a direct probability. Thus, the probability can be calculated through integration over a range of particle velocity. This means that the probability to find a particle at one particular velocity is 0 m/s as the integral will have the same two bounds of integration. Because of this, many statisticians plot cumulative density functions which represent the area under the curve of the probability density function up to a given value of velocity.

[*] Email: rafizadehn@ku.edu

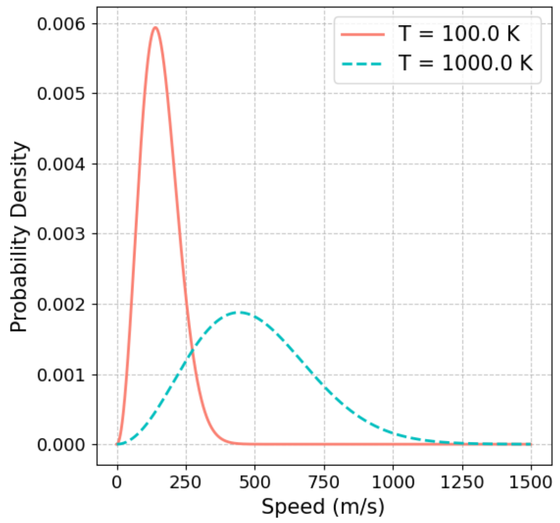

FIG. 1. Probability density for gas particle velocity at differing temperatures, 100 k and 1000 K. Values taken for a gas with an 85 amu molecular mass.

III. CODE AND EXPERIMENTAL SIMULATION

The plots in Fig. 1 are idealized curves that are plotted directly from the M-B distribution equation. To simulate a real experiment sampling gas particles, a random number generator is used in conjunction with the M-B equation and plotted as a histogram, an example is shown in Fig. 2. This is done by defining functions in python to return values of the M-B distribution, and also its inverse as shown in the listing below. From the `rng_MaxBoltz.py` file,

```python
def boltz(x, temp):
    scale = np.sqrt(temp)
    return stats.norm.pdf(x, loc = 0, scale = scale)

def log_likelihood(x, data):
    log_likelihood = 0
    for d in data:
        log_likelihood += np.log(boltz(d, x))
    return -log_likelihood
```

The M-B distribution takes physical parameters as an input and returns a probability density curve, while the log likelihood is calculated at the same time from the generated data. To get an entire distribution, another function can be defined to input arrays of randomly generated numbers to generate random velocities according to the M-B distribution.

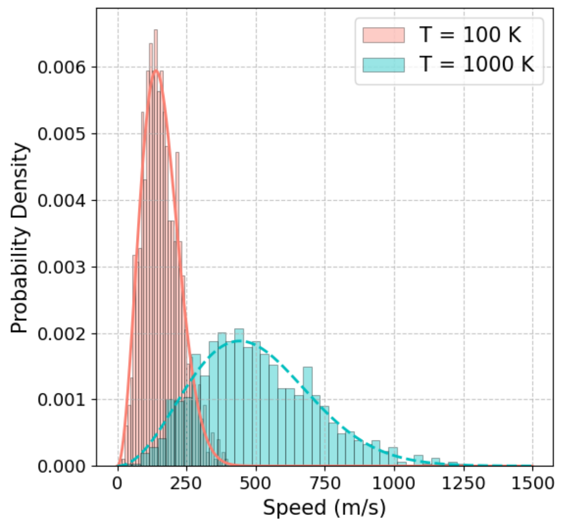

FIG. 2. Histogram of particle velocities taken at a N = 1000 sample size. Values taken for a gas with an 85 amu molecular mass.

The script pair then uses the loglikelihood minimization to estimate the input parameter for each curve.

This section of code uses a python package named `scipy.stats` to calculate the log-likelihood ratios and uses it for the minimization process of the distributions.

```
estimates = []
   for d in data:
       result = optimize.minimize(log_likelihood,
            x0 = 300, args = (d,))
       estimates.append(result.x[0])
```

This makes it easy to graph the estimated values with the true values used in the data generation.

IV. RESULTS AND ANALYSIS

The results of this method are shown in Fig. 3 for an N value of 100 sample sets.

It seems the code works fairly well as it produces the value within a few percent error of the actual value.

V. CONCLUSION

A random number generator was used to sample values from the Maxwell-Boltzmann distribution with a randomly selected set of temperature parameters. Using this

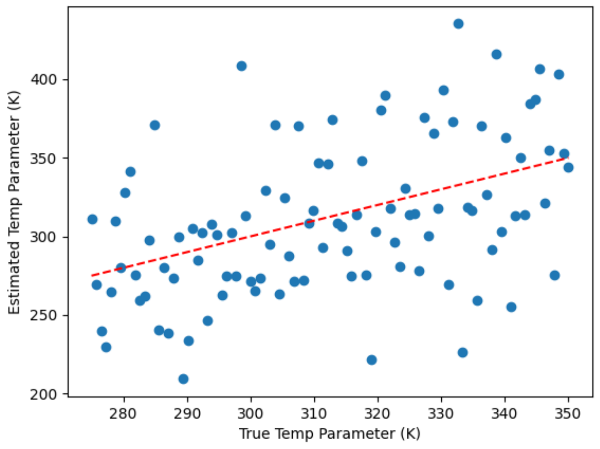

FIG. 3. True temperature parameters plotted against the calculated ones from the log likelihood ratio optimization.

with log likelihood optimization for the experiment, estimated parameter values were able to be tested against the true ones generated by the script. The results show a positive correlation between them with small error compared to the values used.

In the future, this distribution can be studied further by changing other parameters of the M-B distribution. This may include things such as the mass or exploring similar distributions that do not have the same assumptions in their derivation from first principles.

ACKNOWLEDGMENTS

I would like to thank Prof. Chris Rogan for his continued support in the course. I would also like to thank my classmates who assisted me with their peer review.

Appendix A: Sources and Raw Data

All python scripts and data generated are contained in Neema Rafizadeh's GitHub repository for this project. Instructions for running the scripts and changing the inputs are listed on the README.md file listed in that repository.

The `MySort` and `Random` classes of functions can be located at Prof. Christopher Rogan's GitHub. The information regarding the Maxwell-Boltzmann distribution was sourced from the Wikipedia article discussing the subject. The calculation of the statistical power was done with the `statsmodels` package in python, with the docuentation provided here.