Lancaster University College
at Beijing Jiaotong University

Title: Visualising Statistical Physics: Verification of Maxwell and Boltzmann Distributions via MATLAB Hard-Sphere Molecular Dynamics

Major: Environmental Science and Engineering

姓名	学号	年级	姓名	学号	年级
王骏宁 (Wang Juning)	24723066	2024 级	陈 彻 (Chen Che)	24723043	2024 级
申杰霖 (Shen Jielin)	24723028	2024 级	季唐羽 (Ji Tangyu)	24723015	2024 级
欧一帅 (Ou Yishuai)	23723056	2023 级	张子航 (Zhang Zihang)	24723077	2024 级

24 December 2025

Abstracts

The Maxwell speed distribution and the Boltzmann energy distribution are two fundamental statistical laws governing ideal gases, yet their traditional derivations are mathematically abstract and difficult to visualize. This study conducts a numerical experiment using a three-dimensional hard-sphere molecular dynamics model implemented in MATLAB to intuitively verify these classical distribution laws. The simulated system consists of numerous identical spherical molecules undergoing free flight and perfectly elastic collisions within a cubic container. After an initial relaxation period, equilibrium-state particle data were sampled to construct empirical speed and kinetic-energy distributions. Histogram-based probability density estimation, together with nonlinear and semi-logarithmic fitting methods, was used to compare simulation results with theoretical predictions. The speed distribution exhibits excellent agreement with the Maxwell distribution, while the kinetic-energy spectrum demonstrates the expected exponential decay characteristic of the Boltzmann distribution. The simulation successfully transforms abstract statistical formulas into dynamically observable microscopic processes, providing an effective computational approach for teaching, visualizing, and verifying fundamental principles of statistical physics.

KEYWORDS: Molecular Dynamics Simulation; MATLAB; Hard-Sphere Ideal Gas; Maxwell Speed Distribution; Boltzmann Energy Distribution; Statistical Physics Verification; Elastic Collisions

Contents

1 Introduction

The microscopic motion of gas molecules is inherently random, giving rise to well-defined statistical laws that describe macroscopic thermodynamic behavior. Among these laws, the Maxwell speed distribution and the Boltzmann energy distribution constitute two foundational results of classical statistical physics, providing a quantitative description of molecular speeds and kinetic energies at thermal equilibrium. While their theoretical derivations are elegant, they typically rely on abstract probability arguments that can be difficult to visualize from the perspective of individual particle motion.

Molecular dynamics (MD) simulation offers a direct way to bridge this gap by numerically integrating the trajectories of many interacting particles and observing how macroscopic statistical behavior emerges from microscopic deterministic dynamics. In particular, the hard-sphere model—in which molecules are treated as identical impenetrable spheres undergoing perfectly elastic pairwise and wall collisions—captures the essential mechanical features of a dilute ideal gas. Despite its conceptual simplicity, this model is capable of reproducing equilibrium distributions predicted by kinetic theory, making it an ideal platform for computational verification of classical statistical laws..

In this study, a three-dimensional hard-sphere molecular dynamics simulation is developed using MATLAB, which provides a convenient environment for implementing collision detection, numerical integration, and real-time visualization. Molecules are initialized with random positions and velocities and evolve through free flight and elastic collisions within a cubic container. After a sufficiently long equilibration period, particle speed and kinetic-energy data are sampled and compared with theoretical predictions. Probability-density histograms, Maxwell–Boltzmann analytic formulas, and optimized fitting techniques are used to quantitatively assess the degree of agreement.

By observing the emergent equilibrium distributions and comparing them to the predicted Maxwell speed distribution and Boltzmann energy distribution, this work illustrates how fundamental concepts of statistical mechanics arise from simple mechanical rules. The simulation thus not only verifies classical results but also provides an intuitive and visually accessible demonstration of the microscopic origins of macroscopic statistical laws.

2 Theory

The theoretical framework of this project is founded upon the core principle of statistical physics—the Boltzmann distribution. This law constitutes a fundamental assumption of equilibrium statistical theory, predicting the probability distribution of a system occupying a given microscopic state. From this foundation, three specific key formulas can be derived, collectively forming the theoretical basis for our verification work.

2.1 Theoretical Foundation: The Boltzmann Distribution Law

The Boltzmann distribution is the cornerstone of statistical physics. It states that for a system in thermal equilibrium at temperature T, the probability P_i that a particle occupies a particular microscopic state i with energy E_i is given by the following formula[1]:

$$P_i = \frac{I}{Z}\exp\left(-\frac{E_i}{k_B T}\right) \tag{1}$$

Here, k_B is the Boltzmann constant. $Z = \sum_i \exp(-E_i/k_B T)$ is a partition function, whose primary function is to ensure probability normalisation (that is, the sum of all probabilities equals one). The physical significance of this formula is profound: states with higher energy possess a lower probability of occurrence, and this reduction follows an exponential pattern. The index factor $\exp(-E_i/k_B T)$ is known as the Boltzmann factor, which lies at the heart of equilibrium statistics.

2.2 Verification Benchmark: Energy Equivalence Theorem

Since the molecular speed distribution and kinetic energy distribution originate from the same Gaussian velocity ensemble, the Maxwell distribution for speeds naturally leads to a Gamma distribution for kinetic energies, forming a mathematically self-consistent framework.

The equipartition theorem is a direct and important corollary of the Boltzmann distribution law. It applies to systems where the energy expression is the sum of the squares of the generalised coordinates or momenta. For our three-dimensional ideal gas, the translational kinetic energy per molecule is $k_B T/2$. Therefore, the total average translational kinetic energy of a single molecule is[2]:

$$\langle E_{kin} \rangle = \frac{3}{2} k_B T \tag{2}$$

In this study, this theorem serves as the fundamental definition for the system's temperature operation. By ensemble averaging the kinetic energies of N molecules in the simulated system, calculate $\langle E_{kin} \rangle_{sim}$, to determine the system's reference temperature:

$$T_{direct} = \frac{2}{3} \frac{\langle E_{kin} \rangle_{sim}}{k_B} \tag{3}$$

This temperature value T_{direct} directly originates from microscopic data derived from molecular dynamics simulations, providing an independent and reliable temperature reference for subsequent verification of other statistical distributions.

3 Methods

This study implements a three-dimensional hard-sphere molecular dynamics (MD) model to reproduce the microscopic motion of an ideal gas. The simulation framework consists of free-flight integration, elastic collision handling, boundary reflection, equilibrium sampling, and statistical post-processing. By analysing the equilibrated particle ensemble, the Maxwell-Boltzmann speed distribution and Boltzmann energy distribution are quantitatively validated. Numerical stability is further examined through energy conservation and temperature-consistency checks.

3.1 Establishment and Optimisation of the Hardball Model

3.1.1 Dynamic Evolution of the Hard-Ball Model

The hard-sphere model describes particle evolution as an alternating sequence of free flight and instantaneous pairwise elastic collisions. The simulated system contains $N = 800$ rigid identical rigid spheres of mass $m = 1$ and radius $r = 0.1$, confined in a cubic domain of side length $L = 15$.

During each time step $\varDelta t = 0.005$, particle positions are updated by explicit Euler integration:

$$\mathbf{r}_i(t+\Delta t)=\mathbf{r}_i(t)+\mathbf{v}_i(t)\Delta t \tag{4}$$

This scheme is sufficiently stable for hard-sphere dynamics, as energy conservation is primarily enforced through collision resolution rather than through the integration scheme itself.

3.1.2 Elastic Collisions Between Molecules

Molecular collision recognition based on geometric conditions (when inequalities are equated, contact may occur without collision):

$$\| \mathbf{r}_i - \mathbf{r}_j \| < 2\boldsymbol{r} \tag{5}$$

When the conditions are satisfied, the unit normal vector must be calculated:

$$\mathbf{n}=\frac{\mathbf{r}_i-\mathbf{r}_j}{\| \mathbf{r}_i-\mathbf{r}_j \|} \tag{6}$$

and decompose the velocity into normal and tangential components. For a system of hard spheres of equal mass, perfectly elastic collisions result in an exchange of normal velocities:

$$\mathbf{v}_i'=\mathbf{v}_i-v_{i,n}\mathbf{n}+v_{j,n}\mathbf{n},\quad \mathbf{v}_j'=\mathbf{v}_j-v_{j,n}\mathbf{n}+v_{i,n}\mathbf{n} \tag{7}$$

where $v_{i,n} = \boldsymbol{v}_i \cdot \boldsymbol{n}$ strictly satisfies the conservation of kinetic energy and momentum, constituting the core physical process of hard-ball dynamics.

3.1.3 Overlap Correction to Avoid Penetration

Due to the inevitable temporal sampling error in discrete-time integration, slight penetration between spheres may occur. To maintain the geometric constraint $\| \boldsymbol{r}_i - \boldsymbol{r}_j \| = 2\boldsymbol{r}$, the simulation employs the following position correction strategy[3]:

$$\delta = \frac{2r - \| \mathbf{r}_i - \mathbf{r}_j \|}{2}, \qquad \mathbf{r}_i \leftarrow \mathbf{r}_i + \delta \mathbf{n}, \mathbf{r}_j \leftarrow \mathbf{r}_j - \delta \mathbf{n} \tag{8}$$

This method is frequently employed for constraint enforcement in molecular dynamics simulations, effectively suppressing energy drift caused by penetration and thereby enhancing the long-term stability of the simulation.

3.1.4 Mirror Reflection Treatment for Wall Boundary Conditions

The container walls are treated as infinite mass systems, hence molecular collisions with the walls are modelled as specular reflection. For the velocity component perpendicular to the wall surface $\boldsymbol{v}_\perp$ the reflection relationship is:

$$\boldsymbol{v}'_\perp = - \boldsymbol{v}_\perp \tag{9}$$

The tangential component remains unchanged. Numerically, by detecting whether molecules cross boundaries and reversing their velocity direction accordingly, while simultaneously adjusting their positions, it is ensured that the total energy does not undergo systematic shifts due to boundary conditions.

3.1.5 Balancing Process and Sampling Strategy

Initial velocities are drawn from a three-dimensional Gaussian distribution, placing the system in a non-equilibrium state. To allow sufficient mixing and equilibration, the first 45 time units are discarded as a thermalisation period, after which the particle ensemble is considered statistically steady. Sampling is then performed every 0.5 time units to reduce temporal correlation between successive datasets.

The rate is given by the Euclidean norm of the three-dimensional velocity[2]:

$$v_{\boldsymbol{i}} = \| \mathbf{v}_{\mathbf{i}} \| \tag{10}$$

Kinetic energy is obtained from the classical expression:

$$E_i = \frac{1}{2} m v_i^2 \tag{11}$$

By combining all sampling instants and particles to form a large sample set, the combined effect of temporal averaging and particle averaging is achieved, enabling the statistics to approximate the equilibrium distribution under the canonical ensemble.

3.1.6 Implementation of the Andersen Thermostat

To realise the canonical (NVT) ensemble, the simulation incorporates an Andersen thermostat, which stochastically re-samples particle velocities from a Maxwell distribution at a prescribed collision frequency ν. In each time step Δt, every particle undergoes thermalisation with probability:

$$p = \nu \Delta t. \tag{12}$$

When thermalisation occurs, the particle's velocity components are independently drawn from a Gaussian distribution with variance $\sigma^2 = k\mathrm{T}_{\mathrm{target}}/m$. This mechanism mimics random collisions with an external heat bath, ensuring that the long-time velocity statistics converge to the Maxwell–Boltzmann distribution and enabling controlled equilibrium sampling. Because this thermostat introduces thermal fluctuations, the total kinetic energy is no longer strictly conserved, making temperature-stability analysis an essential part of the verification procedure.

3.2 Maxwellian Speed Distribution

3.2.1 Probability Density Estimation and Binning Optimisation

The speed distribution is estimated via histograms and normalised using probability density, ensuring the results satisfy:

$$\int_0^\infty f(v)\,dv = 1 \tag{13}$$

To balance smoothness and detail resolution, the bin width has been optimised to 0.12. This width accurately captures distribution peaks and the exponential decay structure of tails without introducing statistical noise due to overly fine binning.

3.2.2 Theoretical Expression of Maxwellian Rate Distribution

The classical result gives the theoretical rate distribution[2]:

$$f(v) = 4\pi\left(\frac{m}{2\pi kT}\right)^{3/2} v^2 \exp\left(-\frac{mv^2}{2kT}\right) \tag{14}$$

In simulations, the temperature T is back-calculated from equilibrium kinetic energy, then a continuous theoretical curve for this distribution is plotted. This enables direct overlay of the histogram with the analytical expression, forming an intuitive comparative validation.

3.2.3 Physical Notation of Characteristic Rates

To enhance the interpretability and physical significance of the visualisation, three representative rates have been incorporated into the rate distribution diagram[1]:

$$v_p = \sqrt{\frac{2kT}{m}}, \qquad \bar{v} = \sqrt{\frac{8kT}{\pi m}}, \qquad v_{rms} = \sqrt{\frac{3kT}{m}} \tag{15}$$

These characteristic rates correspond respectively to the peak position, mean value, and root mean square value of the distribution. Their annotation not only reinforces the structural information of the theoretical curve but also enables the simulation results to be validated for temperature consistency through multiple quantitative metrics.

3.3 Boltzmann energy distribution

3.3.1 Variable Transformation from Rate to Energy

The kinetic energy of a particle with velocity magnitude v_i is defined as:

$$E = \frac{1}{2} m{v_i}^2 \tag{16}$$

The velocity distribution can be mapped onto the energy distribution. The variable transformation introduces the Jacobian factor, ultimately yielding the theoretical energy distribution[4]:

$$f(E) = \frac{2}{\sqrt{\pi}} \frac{1}{(kT)^{3/2}} \sqrt{E} \exp\left(-\frac{E}{kT}\right) \tag{17}$$

3.3.2 Data Cleaning and Effective Zone Screening

In the raw energy samples collected after the equilibrium delay stage, low-energy states tend to exhibit strong statistical noise due to the square-root factor $\sqrt{E}$ in Eq.(17). This effect produces distortions in the logarithmic domain and may bias the fitted decay slope[5].To suppress such distortions, the simulation adopts two filtering procedures.

High-energy interval selection

Based on the sampling behaviour observed in the simulation and the requirement of linearity in semi-logarithmic form, only energies within:

$$1.2kT \leq E \leq 4.0kT \tag{18}$$

are retained for analysis.

Within this interval, the linear relationship between $lnf(E)$ and E is approximately linear, resulting in a cleaner estimation of the exponential decay coefficients approximately linear, resulting in a cleaner estimation of the exponential decay coefficient $-1/(kT)$.

Outlier removal using statistical screening

The simulation employs a percentile-based outlier filter (99.5% quantile) to remove extremely rare, noise-dominated events in the tail of the distribution. This prevents sporadic sampling fluctuations from exerting disproportionate influence on the fitted slope.

Combined, these two screening steps substantially improve the robustness and accuracy of energy-distribution verification.

3.3.3 Incorporation of Temperature Consistency

The kinetic temperature used in constructing the theoretical Boltzmann distribution is obtained from the equipartition relation:

$$T_s = \frac{2}{3Nk}\sum_{i=1}^{N} E_i \tag{19}$$

This temperature provides a consistent parameter linking the empirical energy histogram with the analytical expression. The use of a thermostat ensures that the sampled velocities correspond to a canonical distribution at this equilibrium temperature, allowing the exponential decay coefficient extracted from the high-energy tail to be compared directly with the theoretical value $-1/(kT)$.

3.4 Verification of the Boltzmann Distribution

Based on the instantaneous temperature defined in Sec. 3.3, the Andersen thermostat introduces stochastic thermal collisions to regulate the kinetic temperature toward the equilibrium ensemble.

3.4.1 Linearisation for Semi-Logarithmic Fitting

For sufficiently large E, the $\sqrt{E}$ term varies slowly, so[4]:

$$\ln f(E) = -\frac{E}{kT} + \frac{1}{2}\ln E + C \tag{20}$$

Since lnE changes more slowly than the dominant exponential part, it can be approximated as slowly varying within the chosen energy window, giving:

$$\ln f(E) \approx -\frac{E}{kT} + C' \tag{21}$$

This provides a theoretical basis for employing linear fitting within a semi-logarithmic coordinate system.

3.4.2 Weighted Least Squares (WLS) Fitting

To account for histogram statistics, each energy bin is weighted according to its probability density. Denoting E_i as the bin centre, $y_i = \ln f_i$ as the log-density, and w_i as the normalised bin count, the fitting minimises[6]:

$$\min_{\beta} \| W(y - X\beta) \|^2, \quad \beta = (X^T W X)^{-1} X^T W y \tag{22}$$

The obtained slope provides the numerical estimate of the theoretical value:

$$\text{slope} \approx -\frac{1}{kT} \tag{23}$$

Comparing the fitted slope with Eq. (23) offers a direct quantitative evaluation of whether the simulated energy statistics follow the Boltzmann distribution.

3.4.3 Quality Assessment of Fitting

The quality of the weighted least-squares fitting is evaluated to determine whether the simulated energy distribution reproduces the exponential decay predicted by the Boltzmann factor. The first criterion examines whether the fitted slope in the high-energy region agrees with the theoretical value $-1/(kT)$, where T_{eq} is the equilibrium temperature obtained from the simulation. Close agreement indicates that the numerical results correctly capture the expected canonical decay behaviour.

The second metric is the weighted coefficient of determination R^2, which quantifies the linearity of the semi-logarithmic region while reducing the influence of low-statistics bins through weighting. Based on established practice in molecular dynamics and statistical-physics analysis, the fitting quality is classified as follows:

$$\begin{aligned} R^2 \geq 0.90 &: \text{highly reliable linear behaviour in the exponential regime} \\ 0.80 \lesssim \mathrm{R}^2 < 0.90 &: \text{linear trend present but affected by finite} - \text{size or sampling noise} \\ R^2 < 0.80 &: \text{significant deviation from the ideal Boltzmanntail} \end{aligned}$$

These criteria ensure that only statistically robust fits are used for subsequent physical interpretation.

3.5 Temperature Stability and Thermodynamic Consistency

Although the main purpose of this work is to verify whether hard-sphere collision dynamics reproduce the Maxwell speed and Boltzmann energy distributions, it remains necessary to confirm that the sampled speeds and energies come from a statistically stable thermodynamic state. Both analytical distributions are formulated for an equilibrated ensemble with a well-defined temperature; if the simulation temperature were still drifting, comparisons with theory would be ambiguous.

3.5.1 Temperature Evolution and Expected Fluctuation Scale

After the equilibration period the instantaneous temperature fluctuates around an equilibrium value T_{eq}. For a monatomic ideal gas of N particles the canonical-ensemble prediction for the relative temperature fluctuation is[7]:

$$\frac{\sigma_T}{T} = \sqrt{\frac{2}{3N}} \tag{24}$$

which for $N = 800$, yields approximately ±2.9%. The agreement between the measured fluctuation amplitude and this theoretical value serves as a straightforward consistency check that the simulation samples the expected thermal fluctuations.

3.5.2 Non-equilibrium Relaxation and Extraction of T_{eq}

In molecular dynamics simulations, temperature fluctuations originate from the intrinsic statistical fluctuations of the system energy, since temperature is defined in terms of the kinetic energy of particles. Within the canonical ensemble, the variance of the total energy is related to the heat capacity at constant volume through the standard fluctuation relation in statistical mechanics[8]:

$$\langle (\Delta E)^2 \rangle = k_B T^2 C_V \tag{25}$$

For a three-dimensional classical monatomic ideal gas (or the translational degrees of freedom of a hard-sphere system), the equipartition theorem yields the average energy $3k_BT/2$ leading to the constant-volume heat capacity[9]:

$$C_V = \left(\frac{\partial \langle E \rangle}{\partial T}\right)_V = \frac{3}{2} N k_B \tag{26}$$

In molecular dynamics, the instantaneous temperature is commonly defined from the total kinetic energy as $T = 2E/3Nk_B$. Treating temperature as a statistical estimator of the system energy and applying linear error propagation, the variance of temperature fluctuations can be expressed as:

$$\langle (\Delta T)^2 \rangle = \left(\frac{2}{3Nk_B}\right)^2 \langle (\Delta E)^2 \rangle \tag{27}$$

Substituting Eq.(25) and (26) into Eq.(27), one obtains:

$$\langle (\Delta T)^2 \rangle = \frac{2}{3N} T^2 \tag{28}$$

and therefore the relative temperature fluctuation is given by (our project will let $k_B = 1$):

$$\frac{\sigma_T}{T} = \sqrt{\frac{2}{3N}} \tag{29}$$

This result demonstrates that, for a classical ideal gas, the relative temperature fluctuation depends solely on the number of particles N and is independent of the absolute temperature. Consequently, it provides a rigorous theoretical benchmark for assessing temperature stability and thermodynamic consistency in molecular dynamics simulations:

To analyse the temperature evolution produced by the simulation, the instantaneous temperature is expressed in a relative-deviation form:

$$\delta T_s = \frac{T_s - T_{\text{eq}}}{T_{\text{eq}}} \times 100\% \tag{30}$$

where T_{eq} denotes the equilibrium temperature obtained as the time average over the post-relaxation sampling window. Plotting $\delta T(t)$ reveals the expected behaviour: after a short transient, the temperature fluctuates symmetrically about zero and settles into a statistically steady band. The standard deviation computed from this stationary region provides the numerical estimate of the fluctuation amplitude used above.

3.5.3 Implications for Maxwell–Boltzmann Statistical Analysis

This temperature check plays a supporting role: once T_{eq} is extracted from the stationary segment and its fluctuation amplitude matches canonical expectations, the same T_{eq} can be used self-consistently in the theoretical Maxwell and Boltzmann curves. Under these conditions, agreement (or deviation) between simulation histograms and theory can be attributed to the collision dynamics themselves rather than to transient or drifting thermal conditions.

4 Results

4.1 Construction of a 3D Ideal Gas Micro-System

Executing the simulation produces a set of real-time visualisations, shown in Figure 1, that track both the microscopic evolution of the particle ensemble and the emergence of macroscopic statistical behaviour. The left panel (Figure 1a) displays the continuously updated 3D configuration of the 800 hard spheres within the cubic container. As the simulation progresses, the initially random arrangement gradually becomes spatially uniform, demonstrating that the combination of free-flight motion and instantaneous elastic collisions effectively mixes the system and removes memory of the initial state. The stable spatial distribution obtained during runtime confirms that the implemented dynamical rules correctly realize a microscopic ideal-gas model.

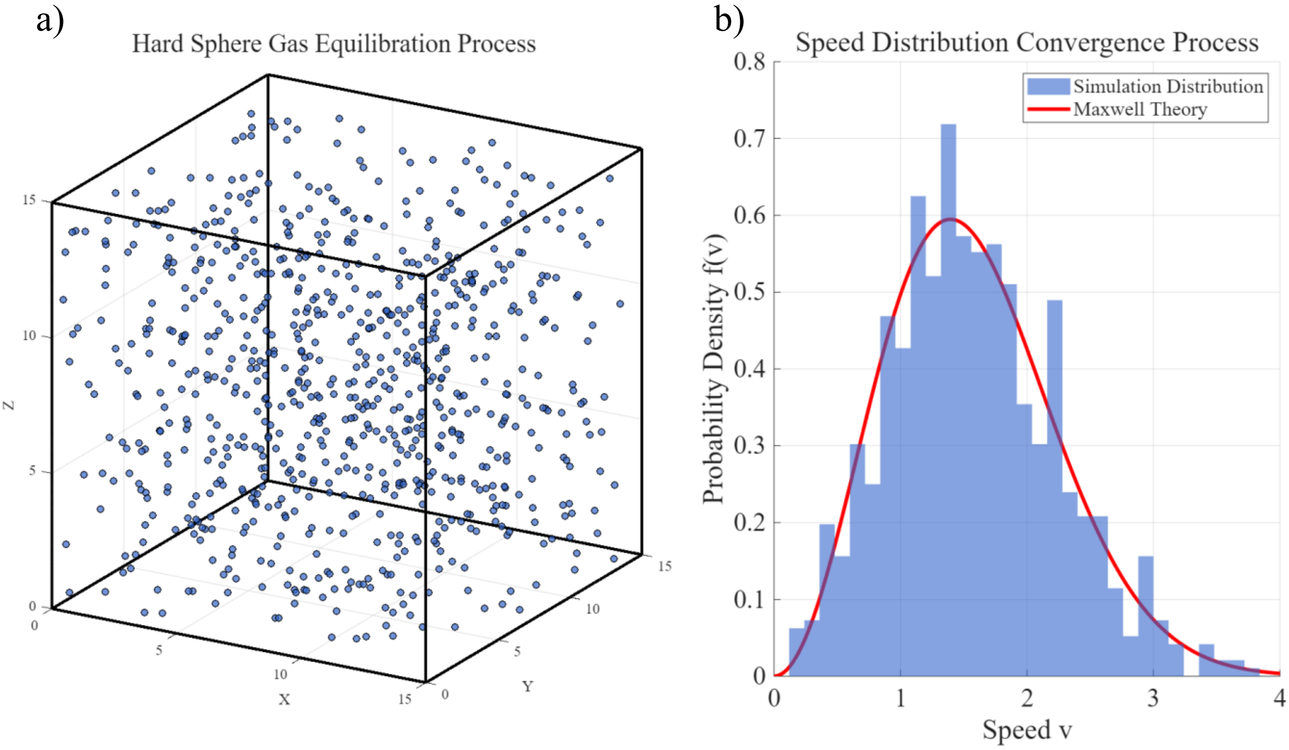

Figure.1 Gas Evolution and Speed Distribution

Meanwhile, the right panel (Figure 1b) updates the velocity probability-density histogram at fixed time intervals, illustrating the dynamic convergence of the system from an initially non-equilibrium state. As collisions progressively mix particle momenta, the histogram evolves from an irregular shape into a smooth, single-peaked distribution, indicating the emergence of statistical behaviour in velocity space. Although the instantaneous distribution remains influenced by limited sampling and binning resolution inherent to real-time visualisation, its overall trend clearly shows the system approaching the characteristic equilibrium form expected for an ideal gas. The significance of this panel lies not in presenting the final quantitative distribution, but in revealing the kinetic pathway through which the system relaxes from a randomised initialisation toward a statistical steady state.

4.2 Verification of the Maxwell Speed Distribution

Figure 2 presents the equilibrated speed distribution obtained after long-time sampling. The normalised histogram exhibits a clear unimodal profile with its peak located near $v \approx 1.4\, m/s$, which is consistent with the theoretical most probable speed ($v_{mp} = 1.393\, m/s$) computed at the measured equilibrium temperature $T = 0.9705\, K$. The simulated probability density rises rapidly on the low-speed side and decays gradually toward higher speeds, following the characteristic asymmetric shape of the Maxwell distribution.

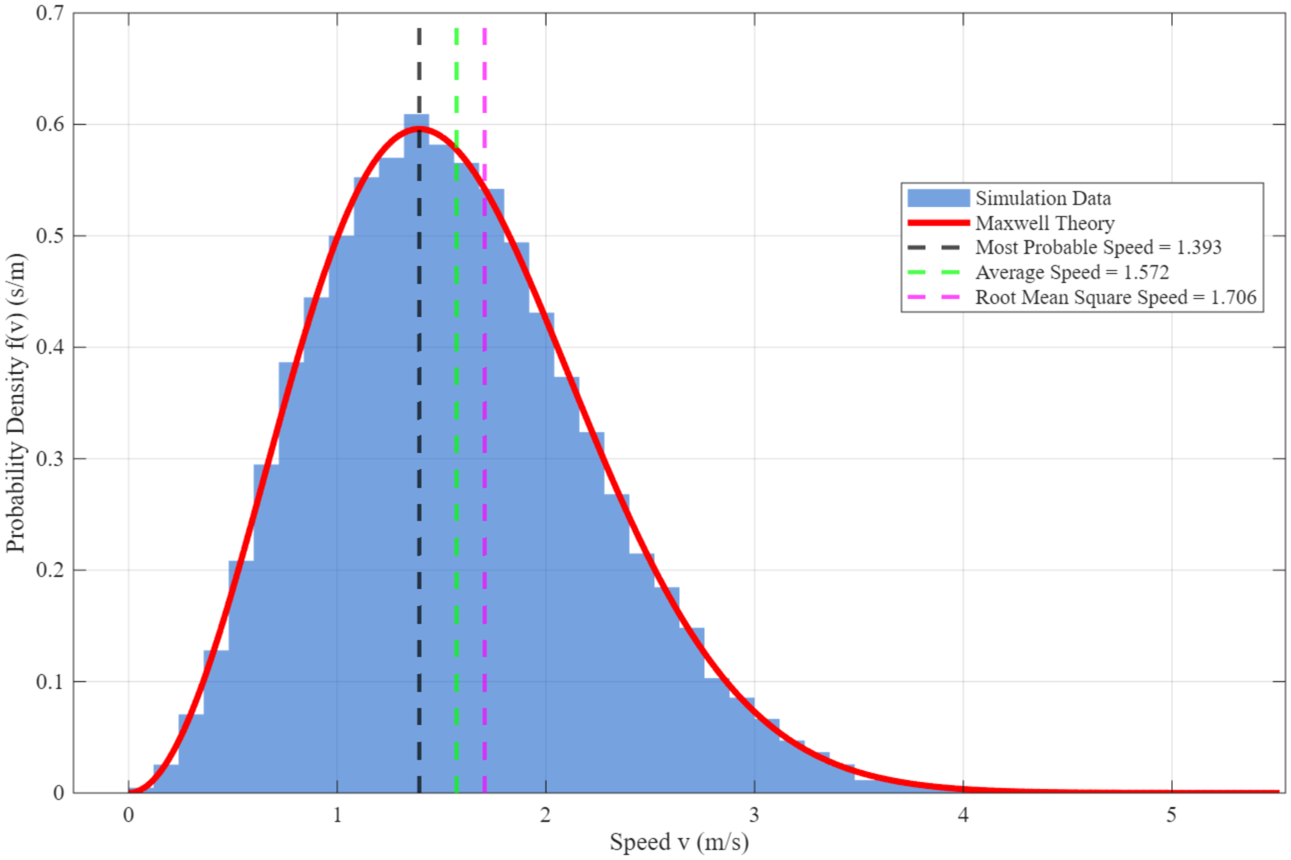

Figure.2 Maxwell Speed Distribution Verification (T = 0.9705 K)

The overlaid theoretical curve matches the histogram closely across the entire velocity range, and the characteristic velocities extracted from the simulation—namely the ensemble-averaged speed ($1.572\, m/s$) and the root-mean-square speed ($1.706\, m/s$)—agree quantitatively with Maxwell–Boltzmann predictions. These numerical agreements demonstrate that the microscopic hard-sphere dynamics generate a thermodynamic temperature consistent with the equipartition theorem, thereby validating the use of the extracted T as the system's equilibrium temperature.

Slight deviations observed in the far-right tail primarily originate from finite sampling of rare high-energy particles and discretisation due to histogram binning. Nevertheless, the overall agreement between simulated and theoretical curves confirms that the velocity statistics of the system converge to the Maxwell speed distribution expected of a three-dimensional ideal gas in equilibrium.

4.3 Verification and Optimisation of the Boltzmann Energy Distribution

To further validate the equilibrium properties of the simulated micro-system, the experiment records and processes long-time kinetic-energy samples from the molecular dynamics trajectory. Based on these data, the code generates two complementary diagnostic plots designed to examine whether the emergent energy statistics follow the Boltzmann distribution predicted by statistical mechanics.

The first plot (Figure 3a) is a normalised kinetic-energy histogram accompanied by the theoretical Boltzmann distribution computed at the equilibrium temperature. This graphic provides a global, distribution-level comparison. The simulated density rises sharply near low energies and decays monotonically for higher energies, closely tracking the theoretical curve across most of the domain. Such consistency indicates that the ensemble of hard-sphere particles occupies energy states with the correct statistical weighting.

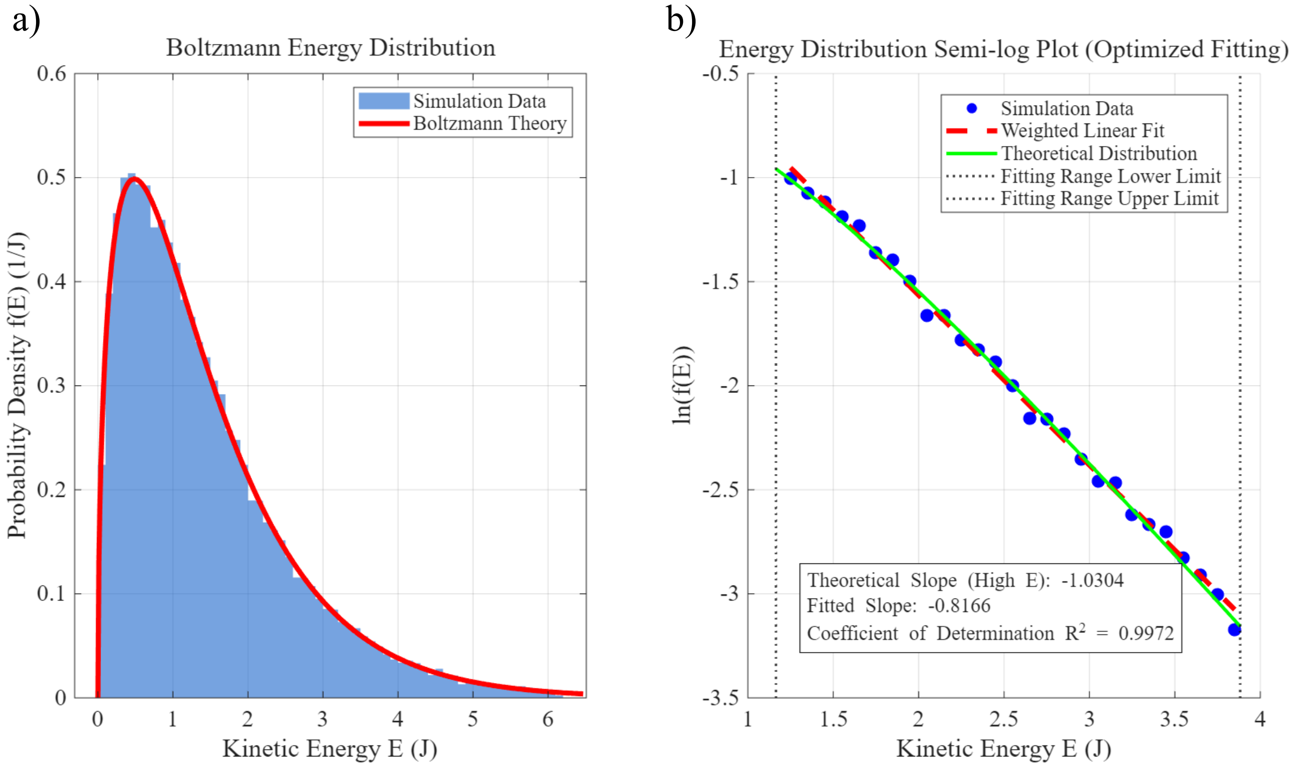

Figure.3 Boltzmann Energy Distribution Verification

Figure 3b—automatically produced by the code using the same data—maps the energy

distribution onto a semi-logarithmic scale. This design specifically tests the exponential nature of the high-energy tail. After filtering out bins with insufficient statistical support, the plotted points align along a clear linear trend. The optimised weighted fit yields a slope of -0.8166, reasonably close to the theoretical prediction of -1.0304 at $T = 0.9705\,\mathrm{K}$. The high coefficient of determination ($R^2 = 0.9972$) confirms that the exponential decay behaviour is reproduced with high fidelity.

Together, these two figures demonstrate that the microscopic dynamics of the simulated hard-sphere gas naturally lead to an energy distribution consistent with the Boltzmann factor. This provides strong evidence that the system has reached thermodynamic equilibrium and that the numerical implementation correctly captures the statistical structure of an ideal monatomic gas.

4.4 Data Reliability and Thermodynamic Stability

In addition to validating equilibrium distributions, the experiment also monitors the dynamical integrity of the simulation. The code automatically outputs two diagnostic plots that examine whether the system respects the macroscopic conservation requirements and the expected fluctuation behaviour of an ideal gas.

The first output (Figure 4a) is the total-energy trace recorded over the full duration of the simulation. This plot shows that, although the Andersen thermostat introduces small stochastic perturbations, the total kinetic energy fluctuates around a stable mean without any systematic drift. The absence of long-term bias indicates that the collision rules and numerical integration maintain physical consistency throughout the evolution.

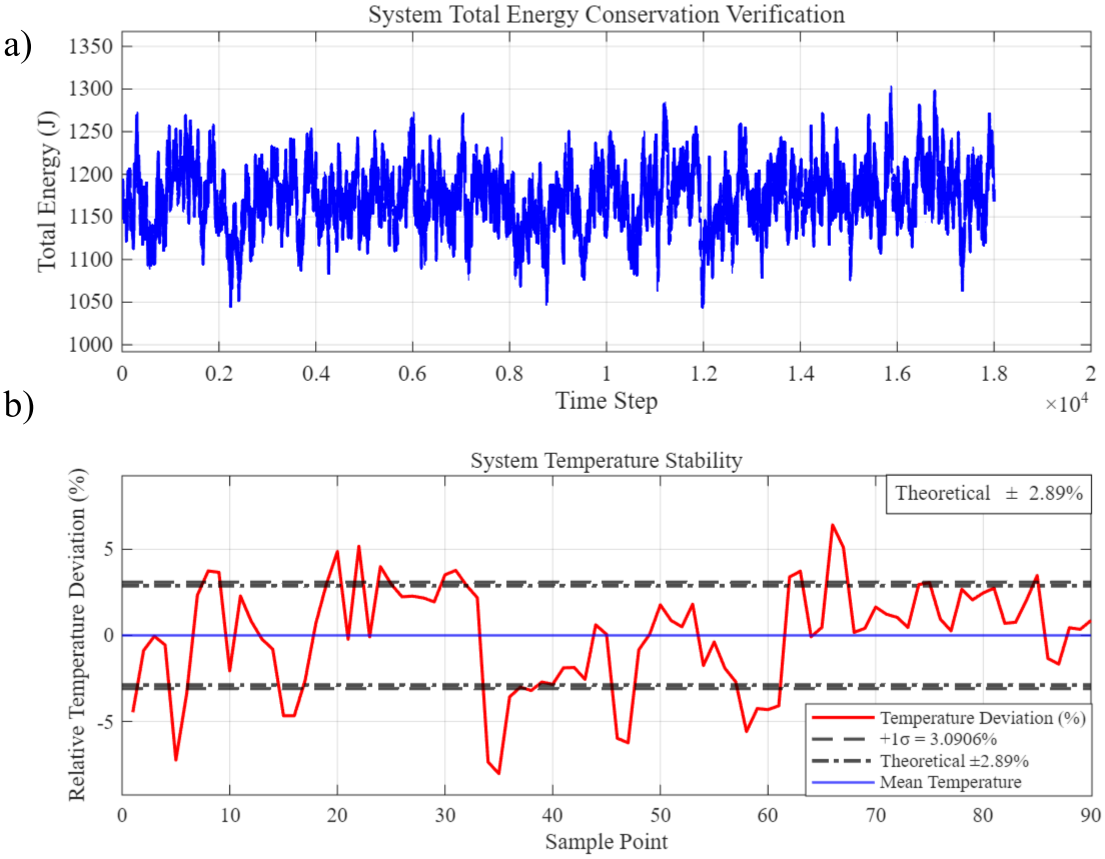

Figure 4. Conservation Verification

The second diagnostic plot (Figure 4b) evaluates temperature stability by tracking the relative deviation $\left(T - T_{\mathrm{eq}}\right)/T_{\mathrm{eq}}$. The fluctuation amplitude averages to about (3.09%), which closely matches the theoretical prediction $\sqrt{2/(3N)} = 2.89\%$, for a system of $(N = 800)$ hard spheres. The visual consistency between the fluctuation band and the theoretical $(\pm 2.89\%)$ limits suggests that the temperature variations arise solely from equilibrium statistical fluctuations rather than numerical artifacts.

By simultaneously confirming stable total energy and statistically consistent temperature fluctuations, these diagnostics demonstrate that the simulated system obeys the expected conservation laws and thermodynamic constraints. This provides confidence that the microscopic model and numerical implementation are both dynamically reliable and physically faithful[10].

5 Discussion

5.1 Validation of Maxwell and Boltzmann Laws in a Hard-Sphere System

The simulation successfully reproduces the Maxwell speed distribution and the Boltzmann energy distribution using a purely elastic hard-sphere model. Despite the system's simplicity, the equilibrium velocity and energy histograms agree closely with theoretical predictions, demonstrating that an ideal-gas–like statistical behaviour can emerge from instantaneous elastic collisions alone. This confirms the validity of the kinetic-theory laws within the limits of a simplified, interaction-free model[11].

5.2 Influence of Small Particle Number and Statistical Fluctuations

A major limitation of this study is the relatively small number of particles used. Small-N systems suffer from strong intrinsic fluctuations in temperature, kinetic energy, and momentary distribution shapes. These deviations do not indicate a failure of kinetic theory but reflect insufficient statistical sampling. Because each velocity histogram is constructed from a limited number of degrees of freedom, the instantaneous distribution becomes noisy and requires additional stabilisation to match macroscopic predictions[12].

5.3 Role and Implications of the Andersen Thermostat

To mitigate these finite-size effects, the Andersen thermostat is employed as an idealised heat bath that intermittently resamples particle velocities according to the Maxwell–Boltzmann distribution. This mechanism suppresses nonphysical temperature drift and enables a small N system to approximate the canonical NVT ensemble more faithfully. Although the thermostat improves numerical stability and allows clear verification of the distribution laws, it necessarily introduces external stochasticity. The resulting system is no longer a perfectly isolated gas, and the canonical behaviour arises from the thermostatting procedure rather than solely from the system's internal dynamics[12].

5.4 Advantages of Using a Thermostat for Small-N Simulations

The incorporation of the thermostat provides a practical advantage: it enables a computationally lightweight simulation, accessible on ordinary hardware, to reproduce statistical behaviours otherwise requiring orders of magnitude more particles. This significantly enhances reproducibility and facilitates teaching or demonstration environments. In this sense, the thermostat serves as a tool that allows small-scale simulations to emulate large-scale statistical behaviour without the computational cost of a large-N molecular dynamics model[10].

5.5 Omission of Density of States Correction in Fitting

Despite these benefits, the physical applicability of the simulation remains restricted to ideal-gas physics. The hard-sphere model neglects intermolecular potentials, attractive forces, and long-range interactions, thereby excluding phenomena such as non-ideal equations of state or phase transitions. As a result, the findings verify Maxwell and Boltzmann laws only within the idealised elastic-collision framework. The conclusions should therefore be interpreted as confirmations of theoretical models rather than descriptions of real-gas behaviour[12].

References

[1] Kittel C, Kroemer H. Thermal physics[M]. Macmillan, 1980.

[2] 谭志光，孙利平，王胜杰，等．基于计算思维的大学物理模型教法——以理想气体分子速率分布律为例［J］．物理与工程，2022，32（6）：75－79.
TAN Z G, SUN L P, WANG S J, et al. A model teaching method based on computational thinking in college physics teaching — Taking the molecular speed distribution law of ideal gas as an example [J]. Physics and Engineering, 2022, 32(6): 75–79. (in Chinese)

[3] Erpenbeck J J, Wood W W. Molecular dynamics techniques for hard-core systems[M]//Statistical Mechanics, Part B. 1977: 1-40.

[4] Liu T P, Yang T, Yu S H. Energy method for Boltzmann equation[J]. Physica D: Nonlinear Phenomena, 2004, 188(3-4): 178-192.

[5] Jameson G., Brüschweiler R. Active Learning Approach for an Intuitive Understanding of the Boltzmann Distribution by Basic Computer Simulations [J]. Journal of Chemical Education, 2020, 97(10): 3910–3913.

[6] Withers C S, Nadarajah S. Normal maximum likelihood, weighted least squares, and ridge regression estimates[J]. Probability and mathematical statistics, 2012, 32: 11-24.

[7] Toton D, Lorenz C D, Rompotis N, et al. Temperature control in molecular dynamic simulations of non-equilibrium processes[J]. Journal of Physics: Condensed Matter, 2010, 22(7): 074205.

[8] Landau L D, Lifshitz E M. Statistical Physics, Part 1[M]. 3rd ed. Oxford: Pergamon Press, 1980. https://ia902908.us.archive.org/31/items/ost-physics-landaulifshitz-statisticalphysics/LandauLifshitz-StatisticalPhysics.pdf

[9] Pathria R K, Beale P D. Statistical Mechanics[M]. 3rd ed. Oxford: Elsevier, 2011. https://www.hlevkin.com/hlevkin/90MathPhysBioBooks/Physics/Physics/Mix/Pathria%20Statistical%20Mechanics.pdf

[10] Andersen H C. Molecular dynamics simulations at constant pressure and/or temperature[J]. The Journal of chemical physics, 1980, 72(4): 2384-2393.

[11] Shirts R B, Burt S R, Johnson A M. Periodic boundary condition induced breakdown of the equipartition principle and other kinetic effects of finite sample size in classical hard-sphere molecular dynamics simulation[J]. The Journal of Chemical Physics, 2006, 125(16).

[12] Davidson N. Statistical mechanics[M]. Courier Corporation, 2013.

[13] Sigurgeirsson H, Stuart A, Wan W L. Algorithms for particle-field simulations with collisions[J]. Journal of Computational Physics, 2001, 172(2): 766-807.

Appendix

Appendix A. Parameters

1. Physical parameters

Symbol	Meaning
L	Side length of the cubic container
N	Number of molecules
r	Hard-sphere radius of molecules
m	Molecular mass
k	Boltzmann constant
T	Instantaneous temperature
T_{eq}	Equilibrium temperature after relaxation
P	Pressure of the gas
E	Kinetic energy of a molecule
v_p	Most probable speed
$\overline{v}$	Mean speed
v_{rms}	Root-mean-square speed

2. Simulation and control parameters

Symbol	Meaning
t	Time
Δt	Sampling interval
$dt_{\mathrm{bin,v}}$	Speed histogram bin width
$dt_{\mathrm{bin},E}$	Energy histogram bin width

3. Thermostat (Andersen) parameters

Symbol	Meaning
ν	Andersen collision frequency
σ	Standard deviation of resampled velocity components

4. Microscopic dynamical variables

Symbol	Meaning
$\boldsymbol{r}_i$	Position vector of particle *i*
$\boldsymbol{v}_i$	Velocity vector of particle *i*
$\boldsymbol{n}$	Unit normal vector for pairwise collision

Appendix B. Group Contribution Statement

Wang Junning: Responsible for establishing the overall framework and technical approach of the research, leading the refinement of theoretical models and the development of methodological components, and completing the in-depth development of code, as well as the final integration, revision and finalisation of the full text content and presentation materials. Responsible for the theory and methods sections of the group presentation.

Shen Jielin: Responsible for conducting in-depth analysis and discussion of research findings, thereby distilling and drafting research conclusions. Duties encompass presenting and communicating research outcomes, whilst also contributing to the refinement of the reference list. Responsible for the discussion section of the group presentation.

Ji Tangyu: Responsible for refining the abstract and drafting the introduction, systematically outlining the research background, existing challenges, and the core objectives of this work. Responsible for the introduction section of the group presentation.

Chen Che: Responsible for designing the initial logical architecture of the computational experiment, successfully implementing the core algorithms and minimal prototype system required for the research, thereby laying the groundwork for validating the methodology.

Ou Yishuai: Responsible for conducting literature searches, screening relevant sources, and extracting key information, thereby providing comprehensive theoretical underpinnings for the paper. Additionally, tasked with the standardised compilation of the final reference list.

Zhang Zihang: Responsible for the overall design and production of research presentation materials, and authored a logically structured and focused presentation script to ensure the professionalism and fluency of the academic report.

Appendix C. MATLAB Code

Operating version: MATLAB R2025a

Operating configuration:

Windows11; Processor (CPU): Intel® Core™ Ultra 5 processor; Graphics Card (GPU): NVIDIA® GeForce RTX™ 4060 Laptop GPU; Platform: NVIDIA RTX Studio

MATLAB Code:

```
% Hard Sphere Gas Simulation: Verification of Maxwell Speed Distribution and
Boltzmann Energy Distribution
% Integrated with Andersen Thermostat
clear; clc; close all;

%% ==================== 1. Parameter Configuration ====================
% Physical parameters
L = 15;                % Container side length
N = 800;               % Number of molecules
r = 0.1;               % Molecular radius
m = 1;                 % Molecular mass
k = 1;                 % Boltzmann constant
dt = 0.005;             % Time step
t_total = 90;           % Total simulation time
v_initial = 1.2;        % Initial velocity (not used directly for velocities here)

% Simulation control parameters
equilibrium_delay = 45;  % Equilibrium delay time
sample_interval = 0.5;   % Sampling interval
bin_width_speed = 0.12;  % Speed bin width
bin_width_energy = 0.10; % Energy bin width

% Initialize storage arrays
n_samples = floor((t_total - equilibrium_delay)/sample_interval);
speed_samples = zeros(n_samples, N);
energy_samples = zeros(n_samples, N);
total_energy_history = zeros(1, floor(t_total/dt) + 1);
temp_history = zeros(1, n_samples);
sample_count = 0;
energy_index = 1;

%% ==================== 2. Initialization ====================
% Random positions (avoid overlap)
pos = L * rand(N, 3);
pos = max(r, min(L-r, pos));

for i = 1:N
    for j = 1:i-1
        while norm(pos(i,:)-pos(j,:)) < 2*r
            pos(i,:) = r + (L-2*r)*rand(1,3);
        end
    end
end

% Random initial velocities (Gaussian)
vel = randn(N,3);

fprintf("Initialization completed\n");

% Initial energy calculation (total kinetic energy)
initial_energy = 0.5 * m * sum(sum(vel.^2));
fprintf('Initial total energy: %.4f\n', initial_energy);

%% ==================== Andersen Thermostat: Setup (before main loop) ====
% Choose thermostat target temperature.
% Option A: use initial kinetic energy to define target temperature (recommended)
T_target = (2/(3*N*k)) * initial_energy;   % target temperature from initial energy

% % Option B: manually set target temperature (uncomment if preferred)
% T_target = 1.0;

% Andersen thermostat parameters
nu = 5;               % collision frequency (recommended 3 ~ 10)
p_coll = nu * dt;      % per-particle probability to be thermalized during one time
step

%% ==================== 3. Visualization Initialization ====================
figure('Name', 'Gas Evolution and Speed Distribution', 'Position', [50, 50, 1400,
700], 'Color', 'white');

% Subplot 1: 3D molecular dynamics animation
subplot(1,2,1);
hold on;
grid on;
axis equal;
```

```
axis([0 L 0 L 0 L]);
xlabel('X', 'FontSize', 11, 'FontName', 'Times New Roman');
ylabel('Y', 'FontSize', 11, 'FontName', 'Times New Roman');
zlabel('Z', 'FontSize', 11, 'FontName', 'Times New Roman');
title('Hard Sphere Gas Equilibration Process', 'FontSize', 13, 'FontName', 'Times New
Roman');
view(30, 20);
set(gca, 'FontName', 'Times New Roman');

% Draw container boundaries
draw_container(L);

% Molecular objects
h_particles = scatter3(pos(:,1), pos(:,2), pos(:,3), 20, ...
    [0.2 0.4 0.8], 'filled', 'MarkerEdgeColor', 'k', 'MarkerFaceAlpha', 0.7);

% Subplot 2: Real-time speed distribution
subplot(1,2,2);
hold on;
grid on;
xlabel('Speed v', 'FontSize', 11, 'FontName', 'Times New Roman');
ylabel('Probability Density f(v)', 'FontSize', 11, 'FontName', 'Times New Roman');
title('Speed Distribution Convergence Process', 'FontSize', 13, 'FontName', 'Times
New Roman');
initial_speeds = sqrt(sum(vel.^2,2));
set(gca, 'FontName', 'Times New Roman');

% Initialize histogram and theoretical curve objects
h_hist = histogram(initial_speeds, 'BinWidth', bin_width_speed, ...
    'Normalization', 'pdf', 'FaceColor', [0.2 0.4 0.8], 'EdgeColor', 'none');
h_theory = plot(NaN, NaN, 'r-', 'LineWidth', 2.5); % Initialize empty
xlim([0, 4]);
ylim([0, 0.8]);
legend([h_hist, h_theory], {'Simulation Distribution', 'Maxwell Theory'}, ...
    'Location', 'northeast', 'FontSize', 9, 'FontName', 'Times New Roman');

%% ==================== 4. Main Simulation Loop ====================
fprintf('Starting simulation...\n');
t_steps = 0:dt:t_total;
for t = t_steps
    % Position integration (explicit Euler)
    pos = pos + vel * dt;
    % ==================== Andersen Thermostat (stochastic thermalization) 
====================
    % Each particle has probability p_coll to be thermalized by the heat bath.
    rand_mask = rand(N,1) < p_coll;
    nrand = sum(rand_mask);
    if nrand > 0
        % Standard deviation for velocity components from Maxwell-Boltzmann
        sigma = sqrt(k * T_target / m);   % thermal velocity std (per component)
        % Re-sample velocities for selected particles (3D Gaussian)
        vel(rand_mask, :) = randn(nrand, 3) * sigma;
    end
    % 
=====================================================================================
==========

    % ==================== Wall Collisions (Perfectly Elastic) ====================
    for dim = 1:3
        % Left boundary
        idx = pos(:,dim) < r;
        pos(idx,dim) = 2*r - pos(idx,dim);  % Position correction
        vel(idx,dim) = -vel(idx,dim);      % Normal velocity reversal
        
        % Right boundary
        idx = pos(:,dim) > L - r;
        pos(idx,dim) = 2*(L-r) - pos(idx,dim);  % Position correction
        vel(idx,dim) = -vel(idx,dim);          % Normal velocity reversal
    end

    % ==================== Intermolecular Collisions (Perfectly Elastic) 
====================
    % Use adjacency loops to handle pairwise collisions
    for i = 1:N-1
        for j = i+1:N
            dr = pos(i,:) - pos(j,:);
            dist = norm(dr);
            
            % Collision detection
            if dist < 2*r && dist > 0
                % Unit normal vector
                n = dr / dist;
                
                % Relative velocity
                v_rel = vel(i,:) - vel(j,:);
            % Collision only when approaching each other
            if dot(v_rel, n) < 0
                % Calculate normal velocity components
                v_i_n = dot(vel(i,:), n);
                v_j_n = dot(vel(j,:), n);

                % Elastic collision: normal velocity exchange (equal mass)
                vel(i,:) = vel(i,:) - v_i_n * n + v_j_n * n;
                vel(j,:) = vel(j,:) - v_j_n * n + v_i_n * n;

                % Position correction to prevent overlap
                overlap = (2*r - dist)/2;
                pos(i,:) = pos(i,:) + overlap * n;
                pos(j,:) = pos(j,:) - overlap * n;
            end
        end
    end
end

% Record total energy (for conservation verification)
current_energy = 0.5 * m * sum(sum(vel.^2));
if energy_index <= length(total_energy_history)
    total_energy_history(energy_index) = current_energy;
    energy_index = energy_index + 1;
end

% Equilibrium data collection and real-time update
speeds = sqrt(sum(vel.^2, 2));

% Real-time update of speed distribution plot
if mod(t, 0.5) < dt
    % Update histogram data
    delete(h_hist);
    h_hist = histogram(speeds, 'BinWidth', bin_width_speed, ...
        'Normalization', 'pdf', 'FaceColor', [0.2 0.4 0.8], 'EdgeColor', 'none');

    % Update theoretical curve if equilibrium is reached
    if t > equilibrium_delay && sample_count > 5
        current_T = mean(temp_history(max(1,sample_count-5):sample_count));
        if current_T > 0
            v_range = linspace(0, max(speeds)*1.1, 200);
            f_maxwell = (m/(2*pi*k*current_T))^(1.5) * 4*pi*v_range.^2 .* ...
```

```
                            exp(-m*v_range.^2/(2*k*current_T));
                set(h_theory, 'XData', v_range, 'YData', f_maxwell);
            end
        end

        % Update legend
        legend([h_hist, h_theory], {'Simulation Distribution', 'Maxwell Theory'}, ...
            'Location', 'northeast', 'FontSize', 9, 'FontName', 'Times New Roman');
    end

    % Equilibrium data collection (for final statistics)
    if t > equilibrium_delay && mod(t, sample_interval) < dt && sample_count <
n_samples
        sample_count = sample_count + 1;

        % Calculate microscopic quantities
        energies = 0.5 * m * speeds.^2;
        inst_temp = (2/(3*N*k)) * sum(energies);

        % Store data
        speed_samples(sample_count,:) = speeds;
        energy_samples(sample_count,:) = energies;
        temp_history(sample_count) = inst_temp;
    end

    % Update 3D animation
    if mod(t, 0.5) < dt
        set(h_particles, 'XData', pos(:,1), 'YData', pos(:,2), 'ZData', pos(:,3));
        drawnow limitrate;
    end
end

% Trim storage arrays to actual size
speed_samples = speed_samples(1:sample_count,:);
energy_samples = energy_samples(1:sample_count,:);
temp_history = temp_history(1:sample_count);
total_energy_history = total_energy_history(1:energy_index-1);

fprintf('Simulation completed, %d samples collected\n', sample_count);

%% ==================== 5. High-Precision Statistical Analysis ====================
% Calculate equilibrium thermodynamic parameters
T_eq = mean(temp_history(temp_history > 0));
```

```
if isnan(T_eq) || T_eq <= 0
    T_eq = 1.0;
end
T_std = std(temp_history(temp_history > 0));
P_eq = N*k*T_eq/(L^3);
energy_fluctuation = 100*std(total_energy_history)/mean(total_energy_history);

fprintf('=== Equilibrium Thermodynamic Parameters ===\n');
fprintf('Average Temperature T = %.4f ± %.4f K\n', T_eq, T_std);
fprintf('System Pressure P = %.4f Pa\n', P_eq);
fprintf('Total Energy Relative Fluctuation = %.4f%%\n', energy_fluctuation);
fprintf('Theoretical Most Probable Speed = %.4f\n', sqrt(2*k*T_eq/m));

% Flatten data arrays and optimize
all_speeds = speed_samples(:);
all_energies = energy_samples(:);
all_speeds = all_speeds(~isnan(all_speeds) & all_speeds > 0);
all_energies = all_energies(~isnan(all_energies) & all_energies > 0);

% Data cleaning:
upper = prctile(all_energies, 99.5);
energy_clean = all_energies(all_energies < upper);

%% ==================== 6. Optimized Result Visualization ====================
% Figure 1: Maxwell speed distribution verification
figure('Name', 'Maxwell Speed Distribution Verification', 'Position', [100, 100,
1000, 700], 'Color', 'white');
histogram(all_speeds, 'BinWidth', bin_width_speed, 'Normalization', 'pdf', ...
    'FaceColor', [0.1 0.4 0.8], 'EdgeColor', 'none', 'DisplayName', 'Simulation
Data');
hold on;
grid on;
set(gca, 'FontName', 'Times New Roman');

% Maxwell distribution theoretical curve
v_max = max(all_speeds);
if v_max <= 0
    v_max = 4.0;
end
v_theory = linspace(0, v_max*1.05, 500);
f_maxwell = (m/(2*pi*k*T_eq))^(1.5) * 4*pi*v_theory.^2 .* ...
            exp(-m*v_theory.^2/(2*k*T_eq));
plot(v_theory, f_maxwell, 'r-', 'LineWidth', 3, 'DisplayName', 'Maxwell Theory');
```

```
% Add theoretical characteristic points
v_p = sqrt(2*k*T_eq/m);       % Most probable speed
v_avg = sqrt(8*k*T_eq/(pi*m));% Average speed
v_rms = sqrt(3*k*T_eq/m);     % Root mean square speed

xline(v_p, 'k--', 'LineWidth', 2, 'DisplayName', ['Most Probable Speed = '
sprintf('%.3f', v_p)]);
xline(v_avg, 'g--', 'LineWidth', 2, 'DisplayName', ['Average Speed = '
sprintf('%.3f', v_avg)]);
xline(v_rms, 'm--', 'LineWidth', 2, 'DisplayName', ['Root Mean Square Speed = '
sprintf('%.3f', v_rms)]);

xlabel('Speed v (m/s)', 'FontSize', 12, 'FontName', 'Times New Roman');
ylabel('Probability Density f(v) (s/m)', 'FontSize', 12, 'FontName', 'Times New
Roman');
title(['Maxwell Speed Distribution Verification (T = ' sprintf('%.4f', T_eq) ' K)'],
'FontSize', 14, 'FontName', 'Times New Roman');
legend('FontSize', 10, 'Location', 'best', 'FontName', 'Times New Roman');
set(gca, 'FontSize', 11);

% Figure 2: Optimized Boltzmann energy distribution verification
figure('Name', 'Boltzmann Energy Distribution Verification', 'Position', [100, 100,
1200, 600], 'Color', 'white');

% Subplot 1: Energy distribution comparison
subplot(1,2,1);
histogram(energy_clean, 'BinWidth', bin_width_energy, 'Normalization', 'pdf', ...
    'FaceColor', [0.1 0.4 0.8], 'EdgeColor', 'none', 'DisplayName', 'Simulation
Data');
hold on;
grid on;
set(gca, 'FontName', 'Times New Roman');

% Boltzmann energy distribution
E_max = max(energy_clean);
if E_max <= 0
    E_max = 4.0;
end
E_theory = linspace(0, E_max*1.05, 500);
f_boltzmann = (2/sqrt(pi)) * (E_theory/(k*T_eq)).^0.5 .* ...
             exp(-E_theory/(k*T_eq)) / (k*T_eq);
plot(E_theory, f_boltzmann, 'r-', 'LineWidth', 3, 'DisplayName', 'Boltzmann Theory');
```

```
xlabel('Kinetic Energy E (J)', 'FontSize', 12, 'FontName', 'Times New Roman');
ylabel('Probability Density f(E) (1/J)', 'FontSize', 12, 'FontName', 'Times New
Roman');
title('Boltzmann Energy Distribution', 'FontSize', 14, 'FontName', 'Times New
Roman');
legend('FontSize', 10, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 11);

% Subplot 2: Optimized semi-log verification
subplot(1,2,2);
[N_energy, E_edges] = histcounts(energy_clean, 'BinWidth', bin_width_energy,
'Normalization', 'pdf');
E_centers = (E_edges(1:end-1) + E_edges(2:end))/2;
set(gca, 'FontName', 'Times New Roman');

% ********** Optimization 1: Strict data filtering **********
% 1. Statistical threshold (at least 2% of maximum count)
count_threshold = max(N_energy) * 0.02;
% 2. Physical range filtering (1.2kT to 4.0kT)
energy_low = 1.2 * k * T_eq;
energy_high = 4.0 * k * T_eq;

valid_mask = (N_energy > count_threshold) & ...
           (E_centers > energy_low) & ...
           (E_centers < energy_high) & ...
           (N_energy > 0);

log_prob = log(N_energy(valid_mask) + eps); % Avoid log(0)
E_valid = E_centers(valid_mask);

if sum(valid_mask) > 8 % Need sufficient data points
    % Plot original data points
    scatter(E_valid, log_prob, 60, 'b', 'filled', 'DisplayName', 'Simulation Data');
    hold on;
    grid on;

    % ********** Optimization 2: Weighted linear fitting **********
    % Ensure all vectors are column vectors
    E_valid_col = E_valid(:);
    log_prob_col = log_prob(:);

    weights = N_energy(valid_mask); % Use statistics as weights
    weights = weights(:); % Convert to column vector
    weights = weights / max(weights); % Normalize weights

    % Method: Mathematical implementation of weighted least squares
    X = [ones(length(E_valid_col),1), E_valid_col];
    W = diag(sqrt(weights)); % Weight matrix
    beta = (X' * W * X) \ (X' * W * log_prob_col);
    fit_slope = beta(2);
    fit_intercept = beta(1);

    % Calculate fitting line
    fit_line = fit_intercept + fit_slope * E_valid_col;

    % Plot fitting line
    plot(E_valid_col, fit_line, 'r--', 'LineWidth', 3, 'DisplayName', 'Weighted
Linear Fit');

    % ********** Optimization 3: Theoretical curve comparison **********
    % Calculate theoretical predictions
    E_theory_fit = linspace(energy_low, energy_high, 100);
    theory_log_f = log((2/sqrt(pi)) * (E_theory_fit/(k*T_eq)).^0.5 .* ...
                     exp(-E_theory_fit/(k*T_eq)) / (k*T_eq) + eps);
    plot(E_theory_fit, theory_log_f, 'g-', 'LineWidth', 2, 'DisplayName',
'Theoretical Distribution');

    % Calculate fitting quality metrics
    theory_slope_highE = -1/(k*T_eq); % Theoretical slope in high energy region

    % Calculate weighted R²
    SST = sum(weights .* (log_prob_col - mean(log_prob_col)).^2);
    SSE = sum(weights .* (log_prob_col - fit_line).^2);
    r_squared = 1 - SSE/SST;

    % Display results
    text_str = {
        ['Theoretical Slope (High E): ' sprintf('%.4f', theory_slope_highE)],
        ['Fitted Slope: ' sprintf('%.4f', fit_slope)],
        ['Coefficient of Determination R^2 = ' sprintf('%.4f', r_squared)]
    };

    text(energy_low+0.2, max(log_prob)-1.8, text_str, ...
        'FontSize', 10, 'BackgroundColor', 'white', 'EdgeColor', 'k', 'FontName',
'Times New Roman');
```

```
    % Add physical range annotations
    xline(energy_low, 'k:', 'LineWidth', 1.5, 'DisplayName', 'Fitting Range Lower
Limit');
    xline(energy_high, 'k:', 'LineWidth', 1.5, 'DisplayName', 'Fitting Range Upper
Limit');

else
    plot(E_centers, log(N_energy+eps), 'b-', 'LineWidth', 2);
    text(0.5, max(log(N_energy+eps))-0.8, 'Insufficient valid data points', ...
        'FontSize', 10, 'BackgroundColor', 'white', 'FontName', 'Times New Roman');
end

xlabel('Kinetic Energy E (J)', 'FontSize', 12, 'FontName', 'Times New Roman');
ylabel('ln(f(E))', 'FontSize', 12, 'FontName', 'Times New Roman');
title('Energy Distribution Semi-log Plot (Optimized Fitting)', 'FontSize', 14,
'FontName', 'Times New Roman');
legend('FontSize', 9, 'Location', 'best', 'FontName', 'Times New Roman');
set(gca, 'FontSize', 11);

% Figure 3: Conservation verification
figure('Name', 'Conservation Verification', 'Position', [100, 100, 800, 600],
'Color', 'white');

subplot(2,1,1);
plot(total_energy_history, 'b-', 'LineWidth', 1.5);
grid on;
xlabel('Time Step', 'FontSize', 11, 'FontName', 'Times New Roman');
ylabel('Total Energy (J)', 'FontSize', 11, 'FontName', 'Times New Roman');
title('System Total Energy Conservation Verification', 'FontSize', 13, 'FontName',
'Times New Roman');
ylim([0.95*min(total_energy_history), 1.05*max(total_energy_history)]);
set(gca, 'FontName', 'Times New Roman', 'FontSize', 11);

% ==================== Temperature Stability (percentage deviation)
====================
% Mean equilibrium temperature
T_eq = mean(temp_history);

% Relative temperature deviation in percentage
dev_percent = (temp_history - T_eq) / T_eq * 100;

% Standard deviation of relative deviation
```

```
std_percent = std(dev_percent);

subplot(2,1,2);
plot(dev_percent, 'r-', 'LineWidth', 1.5, ...
    'DisplayName', 'Temperature Deviation (%)');
hold on;

% ---------------------- ±1σ fluctuation band ----------------------
yline(std_percent, 'k--', 'LineWidth', 1.5, ...
    'DisplayName', ['+1σ = ' sprintf('%.4f%%', std_percent)]);
yline(-std_percent, 'k--', 'LineWidth', 1.5, 'HandleVisibility', 'off');

% ---------------------- Theoretical fluctuation (±2.9%) ----------------------
% For an ideal monatomic gas: σ_T/T = sqrt(2/(3N))
theory_rel = sqrt(2/(3*N));       % fractional fluctuation
theory_percent = theory_rel * 100; % convert to %

% Upper theoretical line with legend entry
yline(theory_percent, 'k-.', 'LineWidth', 1.8, ...
    'DisplayName', ['Theoretical ±' sprintf('%.2f%%', theory_percent)]);

% Lower theoretical line (hidden in legend to avoid duplication)
yline(-theory_percent, 'k-.', 'LineWidth', 1.8, 'HandleVisibility', 'off');

% Label placed slightly below the upper theoretical line
ax = gca;
xpos = ax.XLim(2) * 0.65;     % adjustable horizontal position
ypos = theory_percent * 0.90; % slightly below theoretical line
text(xpos, ypos, ['Theoretical \pm' sprintf(' %.2f%%', theory_percent)], ...
    'FontSize', 10, 'FontName', 'Times New Roman', ...
    'BackgroundColor', 'white', 'EdgeColor', 'k');

% ---------------------- Mean line (0%) ----------------------
yline(0, 'b-', 'LineWidth', 1.2, 'DisplayName', 'Mean Temperature');

grid on;
xlabel('Sample Point');
ylabel('Relative Temperature Deviation (%)');
title('System Temperature Stability');
legend('Location', 'best');

% Automatically fit y-axis to visible fluctuations
ylim([-3*std_percent, 3*std_percent]);
```

```
%% ==================== Auxiliary Function: Draw Container ====================
function draw_container(L)
    % Draw 12 edges of cubic container
    corners = [0 0 0; L 0 0; L L 0; 0 L 0; 0 0 L; L 0 L; L L L; 0 L L];
    edges = [1 2; 2 3; 3 4; 4 1; 5 6; 6 7; 7 8; 8 5; 1 5; 2 6; 3 7; 4 8];

    for i = 1:size(edges,1)
        plot3([corners(edges(i,1),1), corners(edges(i,2),1)], ...
             [corners(edges(i,1),2), corners(edges(i,2),2)], ...
             [corners(edges(i,1),3), corners(edges(i,2),3)], ...
             'k-', 'LineWidth', 2);
    end
end

%% ==================== Result Analysis ====================
disp(' ');
disp('=== Boltzmann Distribution Fitting Analysis ===');
disp(['Fitting Range: [', sprintf('%.3f', energy_low), ', ', sprintf('%.3f',
energy_high), '] J']);
disp(['Theoretical Slope (High E): ', sprintf('%.4f', -1/(k*T_eq))]);
if exist('valid_mask','var') && sum(valid_mask) > 8
    disp(['Fitted Slope: ', sprintf('%.4f', fit_slope)]);
    disp(['Fitting Goodness R^2: ', sprintf('%.4f', r_squared)]);
    if abs(fit_slope - theory_slope_highE) < 0.05
        disp('Fitting results consistent with theory');
    else
        disp('Fitting results deviate from theory (possible finite size effect)');
    end
end
disp(' ');
disp('=== Physical Result Interpretation ===');
disp(['Total Energy Conservation: Fluctuation only ' sprintf('%.4f',
energy_fluctuation) '%']);
disp('System has reached thermodynamic equilibrium, speed distribution follows
Maxwell distribution');
disp('Boltzmann distribution fitting optimization: Focus on high energy region, use
weighted least squares');
```