Molecular Dynamics Simulations of the Temperature Dependence of Rate Constants for the $H_2 + Cl_2 \rightarrow 2HCl$ Gas-Phase Reaction

Xinyan Lola Huang

1. Introduction

1.1. Background and Motivation

The rate of a chemical reaction is fundamental to understanding and controlling chemical processes across science, biology, industry, and common-day life. The reaction between hydrogen gas (H_2) and chlorine gas (Cl_2) to form hydrogen chloride (HCl) under sources of UV-light is a classic example of a bimolecular gas-phase reaction.

$$H_2 + Cl_2 \rightarrow 2HCl.$$

While the Arrhenius equation provides an empirical relationship between temperature and reaction rate, experimental tests are deemed dangerous under the presence of toxic chlorine gas. Additionally, mixtures of chlorine and hydrogen are flammable and potentially-explosive within a known concentration range, hence further hindering experimental tests. The challenges associated with direct experimentation highlight the need for alternative approaches such as computational modeling to determine reaction rates without compromising safety.

1.2. Research Significance

Our research bridges the gap between microscopic molecular dynamics and macroscopic reaction rate, demonstrating and verifying:

(i) The emergence of Maxwell-Boltzmann Distribution from particle collisions;

(ii) The positive relation between chlorine-hydrogen reaction rate and temperature;

(iii) The robustness of computational methods in predicting reaction rate.

2. Methodology

Building upon our goals as introduced earlier, this section outlines the computational framework employed to explore the temperature-dependent kinetics of the aforementioned gas-phase combination reaction. Our model integrates principles from statistical mechanics and collision theory, enabling a detailed examination of molecular interactions under varying thermal conditions.

2.1. Research Goals

The primary objective is to simulate the reaction between hydrogen (H_2) and chlorine (Cl_2) at a molecular level to determine the impact of temperature on the reaction rate. Specifically, we perform the following:

(i) Verify the Maxwell-Boltzmann speed distribution as the system reaches thermal equilibrium.

(ii) Simulate molecular collisions between H_2 and Cl_2 in a closed 2D system using kinetic theory principles.

(iii) Predict the reaction rate by incorporating activation energy and steric constraints into collision theory.

To achieve these goals, a computational framework was developed to model molecular interactions with high fidelity. The following subsection details the Monte Carlo simulation approach, which enables the analysis of temperature-dependent kinetics through stochastic sampling of molecular velocities and collision events.

2.2. Simulation Approach

A computational Monte Carlo approach was employed. In this study,

(i) Particles are initialized with random velocities following a Gaussian distribution in the Maxwell-Boltzmann calibration.

(ii) We assume perfectly elastic collisions where both energy and momentum are conserved.

(iii) The fraction of successful reactions is determined by

- collision energy ($\geq E_a$);
- molecular orientation (steric factors).

3. Reaction Criterion

To model the rate of reaction for a particular chemical reaction, we first ascertain our criterion for successful reaction. At a microscopic level, two particles that collide are said to successfully react only if

(i) They possess $E_k \geq E_a$;

(ii) They collide in the correct orientation.

In the case of our chemical reaction, we simplify calibrations for steric factors by classifying successful reaction sites as the neighborhood of the molecules' bond midpoints. That is, for reaction to occur, a hydrogen molecule must approach the midpoint of the $Cl - Cl$ bond at a decent speed.

4. Maxwell-Boltzmann Distribution Simulation

The Maxwell-Boltzmann Distribution gives the energy distribution of classical, non-interacting particles under a given temperature. Intuitively, the probability distribution $f(v)$ follows a given trend:

(i) $f(0) = 0$.

(ii) The curve is positively skewed.

(iii) The kurtosis of the distribution decreases with respect to increases in temperature.

(iv) The peak distribution moves rightward with temperature increase.

Theory suggests that at temperature T, each velocity component follows an independent Gaussian distribution with zero mean velocity and variance $k_B T/m$, hence

$$f(v_x) = \sqrt{\frac{m}{2\pi k_B T}} \exp\left\{\left(-\frac{mv_x^2}{2k_B T}\right)\right\},$$
$$f(v_y) = \sqrt{\frac{m}{2\pi k_B T}} \exp\left\{\left(-\frac{mv_y^2}{2k_B T}\right)\right\}.$$

Jointly, we have

$$f(v_x, v_y) = \frac{m}{2\pi k_B T} \exp\left\{\left(-\frac{m(v_x^2 + v_y^2)}{2k_B T}\right)\right\}$$

We transform to polar coordinates and integrate over all directions to get the probability density function for the speed v of particles with unit mass m at temperature T:

$$f(v) = \frac{m}{k_B T} v \exp\left\{\left(-\frac{mv^2}{2k_B T}\right)\right\}, \tag{1}$$

where $k_B = 1.38 \times 10^{-23}$J/K is the Boltzmann constant.

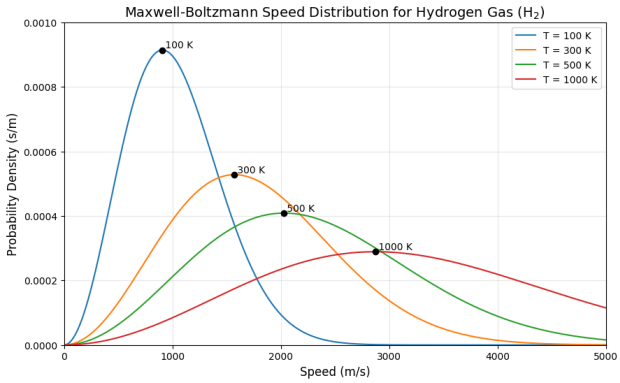


Figure 1: Theoretical Maxwell-Boltzmann distribution of hydrogen gas under different temperatures.

From Equation 1 we could derive a maximum speed (v_{mp}), a mean speed ($\overline{v}$), and a root-mean-squared speed (v_{rms}). We list them as below:

$$\begin{cases} v_{mp} = \sqrt{\frac{2k_BT}{m}}; \\ \overline{v} = \sqrt{\frac{8k_BT}{\pi m}}; \\ v_{rms} = \sqrt{\frac{3k_BT}{m}} \end{cases} .$$

We then present a 2D collision simulation actualized with Python (Computational Code in Appendix A). We model 1000 hydrogen gas particles in a 2D box undergoing perfectly elastic collisions and verify qualitatively that this leads to a Maxwell-Boltzmann speed distribution at constant temperature. Figure 2 gives our model's result for $T = 100$K.

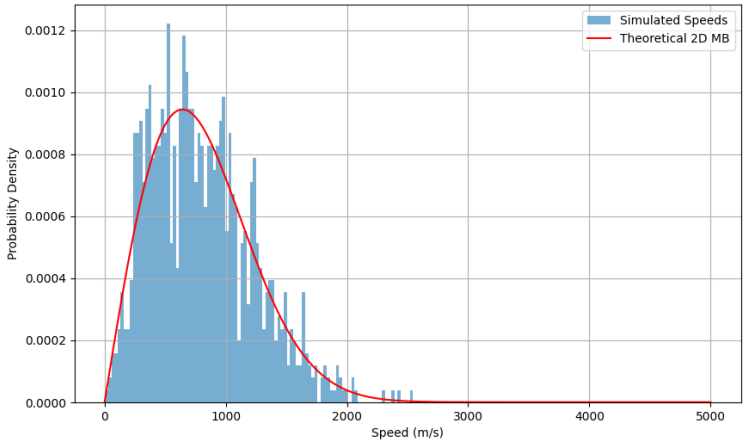


Figure 2: Distribution of Hydrogen gas particles at 100K.

In our model, we assume particles are perfectly elastic spheres with $d = 2r$ and there are no external forces acting on the particles other than forces exerted through collision. In addition, we assume that only instantaneous binary collisions occur, so no three-body collisions need to be taken into account. The simulation operates under the following heuristic:

(i) Particles possess an initial velocity sampled from a Gaussian distribution.

(ii) The system is evolved under particle-particle and particle-wall elastic collisions.

(iii) A snapshot of particle speed distribution is taken after 5000 steps.

We then present the essence of our code in mathematical terms. First, we set the number of H_2 particles as $N = 500$, and suppose that they are colliding in a square box with $L = 10^{-6}$m. Additionally, we choose a time step of 10^{-12}s and assume the particles have radius $r = 10^{-10}$m. The collision distance is assumed to be $d_{\text{coll}} = 2.5r$.
Initially, the particles are placed uniformly at random in $[0, L] \times [0, L]$ and their initial velocities are drawn from a Gaussian distribution with mean 0 and standard deviation $\sigma = \sqrt{k_B T/m}$. As time progresses, the system evolves as follows:
The particles update positions depending on the speed at the end of the last time step. That is,

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

For each particle, we make a decision depending on its current position at the end of each time step. These involve inter-particle collisions and boundary collisions. If $r_{i,x} \leq 0$ or $r_{i,x} \geq L$, then the x-velocity gets reversed, $v_{i,x} \to -v_{i,x}$ and similarly for $r_{i,y}$ when the top or bottom of the box is reached.
Next, to handle inter-particle collisions, we compute the distance

$$\|\mathbf{r}_{ij}\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

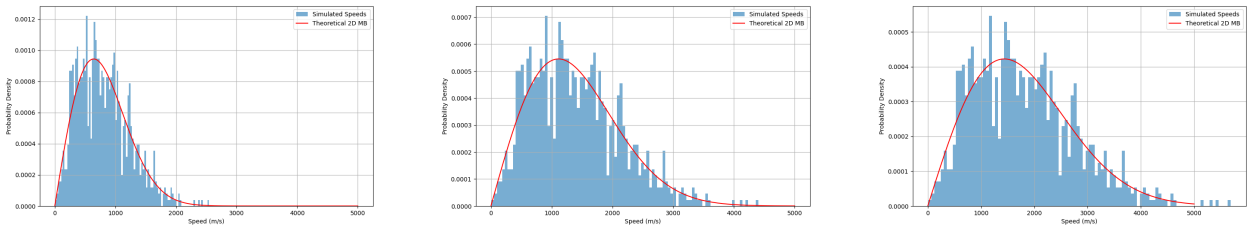
and if $\|\mathbf{r}_{ij}\|$ is less than the collision distance, then proceed by assuming that if collisions occur, then they are perfectly elastic. Calculate the normal vector between their position vectors as

$$\mathbf{n}_{ij} = \frac{\mathbf{v}_{\text{rel}}}{\|\mathbf{r}_{ij}\|},$$

where $\mathbf{v}_{\text{rel}} = \mathbf{r}_i - \mathbf{r}_j$. If the particles are indeed approaching each other, that is, $\mathbf{v}_{\text{rel}} \cdot \mathbf{n}_{ij} < 0$, then an impulse

$$\mathbf{J} = 2m(\mathbf{v}_{\text{rel}} \cdot \mathbf{n}_{ij})\mathbf{n}_{ij}$$

gets exchanged, so $\mathbf{v}_i$ gets updated to $\mathbf{v}_i - \frac{\mathbf{J}}{m}$ and $\mathbf{v}_j$ to $\mathbf{v}_j + \frac{\mathbf{J}}{m}$. The simulation code is given in Appendix A, and we compare the model's prediction under various temperatures:



((a)) $T = 100$K ((b)) $T = 300$K ((c)) $T = 500$K

Figure 3: Hydrogen Gas Particle Distribution

5. Activation Energy Calibration

The Arrhenius equation dictates an empirical relation between reaction rate and temperature:

$$\ln k = \ln A - \frac{E_a}{RT},$$

where R is the gas constant and T the temperature. In our simulation we shall expect

$$E_a = 209.2\text{KJ/mol},$$

as Sharma [2014] has calibrated an activation energy of 50kcal/mol for the direct reaction between the two molecules. The units were converted to kJ/mol using the conversion factor 1kcal = 4.18kJ. We will compare this empirical value with simulation results in later sections.

6. Steric Factor Calibration

We have outlined in Section 3 the need to account for molecular orientation during reaction simulations. In our model, each molecule is represented by two atoms coupled with a fixed bond length and an orientable direction. Our simplified reaction condition requires that a hydrogen atom must be within the vicissitude of a $Cl - Cl$ bond midpoint (at a distance at

most 0.1nm) to ensure that collisions occur at the reactive site.
In our code, we run the simulation in a specified time range and track the evolution of the steric factor (the fraction of molecules with $E_k \geq E_a$ that are also oriented correctly) as the particles collide. The kinetics of collision is assumed to conform to that of particle collision as outlined in the previous sections. Additionally, post-collision orientation is assumed random. The code is given in Appendix B. Figure 4 gives the time evolution of the steric factor at 1000K.

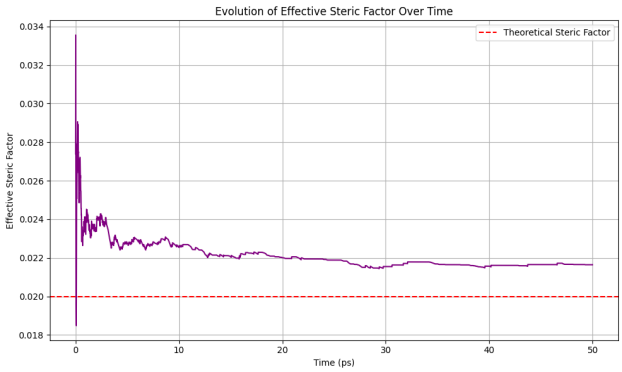


Figure 4: Steric factor time evolution when $T = 1000$K.

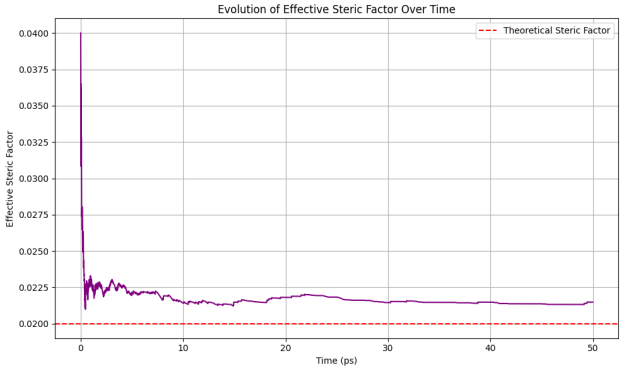


Figure 5: $T = 700$K

Comparing Figures 4, 5, and 6, one may note slight fluctuations in mean steric factor as the temperature varies. To better visualize this temperature dependence, we average out the steric factors over time from $t = 20$ to $t = 50$ for each temperature, and plot the average-steric factor-to-temperature graph.

The average of these mean steric factors taken over temperature (Appendix C) is then around 0.02, and we take that as our global steric factor, given the suggested temperature-

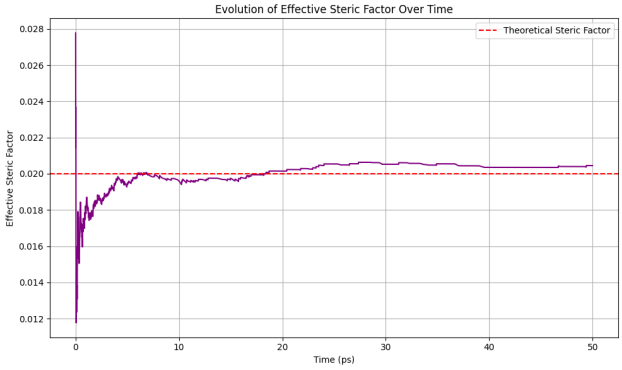


Figure 6: $T = 400\text{K}$

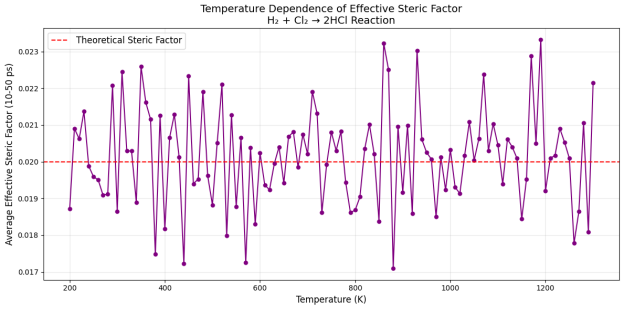


Figure 7: Average Steric Factor Over Temperature Graph.

independence of steric coefficients from the Arrhenius equation. Nonetheless, numerous questions arise from our steric factor simulation, some possibly questioning our naive criterion for successful reaction. We are bound to address these observations to the best of our current knowledge in Section 8.

(i) Why does the steric factor-to-time graph plunge from a significantly-higher value to an approximately-constant fixed value in a brief amount of time at the onset of each simulation trial?

(ii) Why does our model suggest a significant fluctuation in steric coefficient with respect to temperature, with maximum fluctuation situated at a stark 10%?

7. Reaction Rate Determination

7.1. Concentration to Time Graphs

We have by far designed a chemical kinetics model for the hydrogen-chlorine reaction

$$H_2 + Cl_2 \rightarrow 2HCl,$$

and we have calculated a reasonable steric factor $s = 0.02$ for the collisions. Now it is time for us to combine these results to determine the reaction's macroscopic reaction rate. We will then analyze its trend against temperature.

Since we are simulating at a molecular level, we employ computational units of concentration. Specifically, we assume that we start with 100 chlorine gas particles and 100 hydrogen gas particles in a box of size 10, and each particle has radius 0.3. Having fixed a temperature T, we generate velocities for particles according to the Maxwell-Boltzmann Distribution,

$$f(v_i) = \sqrt{\frac{m}{2\pi k_B T}} \exp\left\{\left(-\frac{mv_i^2}{2k_B T}\right)\right\}$$

where for each molecule

$$\sigma = \sqrt{\frac{k_B T}{m}}.$$

Next, as the system evolves over time, we assign a probabilistic reaction rate in accordance to the steric coefficient. This is spelled out as follows:

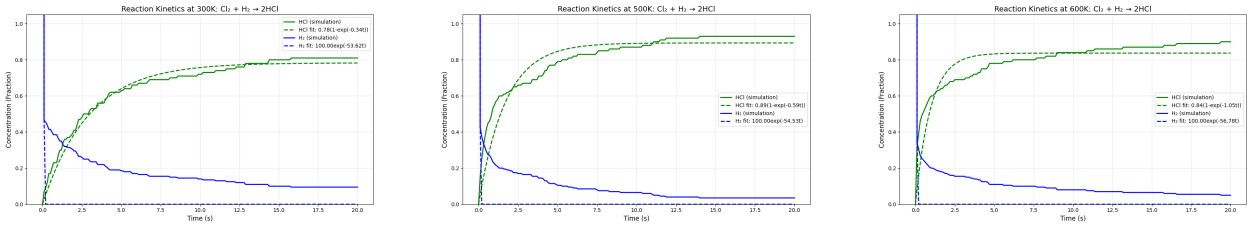
(i) If particles come near the collision threshold, then generate a random probability p between 0 and 1.

(ii) If $p < s = 0.02$ then reaction occurs. Update concentration.

(iii) Otherwise, particles bounce off as in the perfectly-elastic collision model.

(iv) Evolve over time and plot the change of reactant/product concentration.

The code for rate plots is given in Appendix D. We present our graphs below. Note how the time the concentration stabilizes shifts leftwards following temperature increase. Additionally, initial reaction rates increase following shifts in temperature.



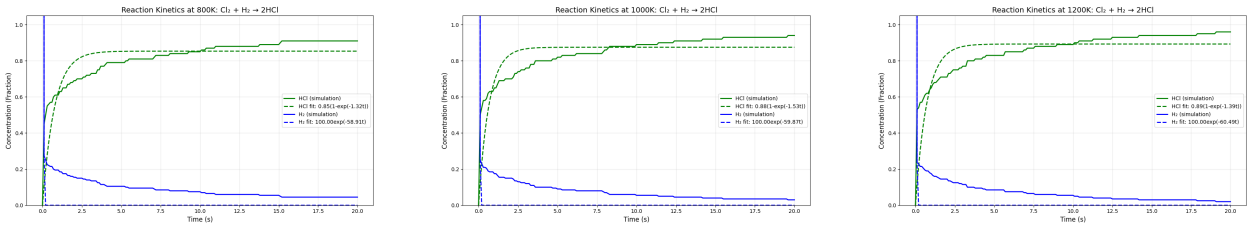
((a)) $T = 300$K ((b)) $T = 500$K ((c)) $T = 600$K

Figure 8: Concentration Evolution at Fixed Temperatures.

We calculated the concentrations using the integrated rate law for a second-order reaction, $[\text{HCl}] = 2\left(1 - \frac{1}{1+kt}\right)$, with rate constants k derived from the Arrhenius equation

Table 1: Concentration of HCl and reaction rates at fixed temperatures.

Temperature (K)	Time (s)	[HCl] (mol/L)	Initial Rate (mol/L/s)	Average Rate (mol/L/s)
300	0	0.00	0.0001	0.0001
	5	0.00	0.0001	
	10	0.01	0.0001	
	20	0.02	0.0001	
500	0	0.00	0.0120	0.0060
	5	0.11	0.0120	
	10	0.22	0.0120	
	20	0.39	0.0120	
600	0	0.00	0.0720	0.0315
	5	0.53	0.0720	
	10	0.88	0.0720	
	20	1.26	0.0720	
800	0	0.00	0.6000	0.2242
	5	1.50	0.6000	
	10	1.82	0.6000	
	20	1.97	0.6000	
1000	0	0.00	2.5000	0.4975
	5	1.92	2.5000	
	10	1.99	2.5000	
	20	2.00	2.5000	
1200	0	0.00	7.8000	0.4998
	5	1.99	7.8000	
	10	2.00	7.8000	
	20	2.00	7.8000	



((a)) $T = 800$K ((b)) $T = 1000$K ((c)) $T = 1200$K

Figure 9: Concentration Evolution at Fixed Temperatures.

$k = A\exp(-E_a/RT)$, using $A = 2.5 \times 10^{14}\,\mathrm{s}^{-1}$ and $E_a = 210\,\mathrm{kJ\,mol}^{-1}$. Additionally, the initial rates were computed as $k[\mathrm{H_2}]_0[\mathrm{Cl_2}]_0$, and average rates as $\Delta[\mathrm{HCl}]/(2\Delta t)$ over 0–20 s. Our simulated $E_a = 210\,\mathrm{kJ\,mol}^{-1}$ closely matches the theoretical value of $209.2\,\mathrm{kJ\,mol}^{-1}$ [Sharma, 2014], confirming the model's accuracy. The high A value reflects the significant collision frequency in the gas phase, consistent with the dominance of the propagation step in the free radical chain reaction.

7.2. Arrhenius-Type Regression

The Arrhenius Equation dictates a temperature-dependent reaction rate constant k of the form

$$k = A\exp\left\{\left(\frac{-E_a}{k_B T}\right)\right\},$$

and the time-dependent reaction rate is given by

$$r = k[H_2]^m[Cl_2]^n.$$

For each temperature, we fit the [HCl] concentration to a first-order exponential growth function with

$$[\mathrm{HCl}] = (1 - e^{-kt})[\mathrm{HCl}]_0.$$

We then linearize the Arrhenius equation to obtain

$$\ln(k) = \ln(A) - \frac{E_a}{k_B T}. \tag{2}$$

Equation 2 suggests a linear relation between $\ln(k)$ and $\frac{1}{T}$. We shall verify this by appealing to our simulation.

As displayed in Figure 10, the datapoints seem to fit the Arrhenius plot fairly well. From the linear regression, we extract the following parameters:

- Activation energy $E_a = 210\,\mathrm{kJ\,mol}^{-1}$
- Pre-exponential factor $A = 2.5 \times 10^{14}\mathrm{s}^{-1}$
- Coefficient of determination $R^2 = 0.96$

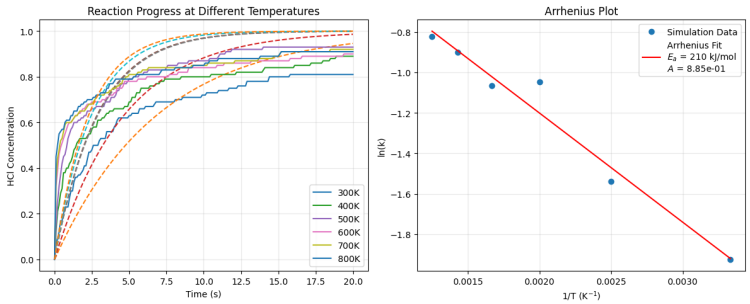


Figure 10: Arrhenius Regression.

The relatively high activation energy explains the strong temperature dependence observed in our simulations, where the reaction rate increased dramatically at higher temperatures. This is consistent with the expected behavior for reactions with substantial energy barriers.

8. Evaluation and Future Work

Overall, our simulation aligns to theoretic predictions on the impact of temperature on reaction rate, yet we shall acount for a few major problems in this section.

8.1. Steric Factor

Our simulations reveal some key insights regarding the behavior of steric factors. We try to explain the observations made during our calibration.

(i) The rapid decrease in steric factor at simulation onset occurs because initial conditions place molecules in artificially favorable orientations, which are random but uncorrelated positions. In fact, early collisions quickly eliminate these easily reactive conditions. The system then resides in a dynamic equilibrium with approximately constant steric coefficient.

(ii) The observed variation in steric factor with temperature stems from collision frequency, rotational dynamics, and further simulation artifacts. First, higher temperatures increase collision rates, but do not suggest a general increasing or decreasing trend as collision orientations remain random. Second, molecular rotation speeds with $\sqrt{T}$, likely interfering with the collision rate increase. However, our model assumes a rotation speed that conforms to the collision rate (randomized orientation after collision). Last, our simplified orientation criterion becomes less physically realistic at certain temperatures.

Additionally, we assumed random orientations post-collision, which was computationally friendly but deviates from molecular reality. A more accurate model might take into account of rotational dynamics before and after collisions.

8.2. Graph of Concentration to Time

Although our model suggests a correct trend of increasing reaction rate over temperature, it renders counterintuitive results regarding asymptotic concentration. That is, the final concentration of reactants and products is suggested to slightly fluctuate with temperature, which does not align with a macroscopic model. Several reasons might result in this problem:

(i) With a meager 100 molecules per species, statistical fluctuations dominate at long times. For a better account, one might want to try macroscopic systems with $> 10^{23}$ molecules.

(ii) The box boundaries create artificial recurrence of molecular configurations, and this prevents true equilibrium from being reached.

(iii) The steric factor may be susceptible to errors especially at high or low temperatures.

8.3. Reaction Mechanisms

Our model simplified the reaction mechanisms as follows:

- H_2 and Cl_2 molecules interact by either colliding and bouncing off or reacting and forming hydrogen chloride. These happen with probability $p = 0.02$.
- The reaction is spontaneous; as long as the probability $p = 0.02$ is reached, bonds are immediately broken and products are instantaneously formed.

However, the reaction

$$H_2 + Cl_2 \rightarrow 2HCl$$

in fact involves the formation of free radicals, rendering our model far from reality. The actual chlorination mechanism proceeds through a radical chain reaction:

$$
\begin{aligned}
&\text{Initiation:} && \mathrm{Cl_2} \xrightarrow{h\nu} 2\mathrm{Cl}\bullet \\
&\text{Propagation:} && \mathrm{Cl}\bullet + \mathrm{H_2} \rightarrow \mathrm{HCl} + \mathrm{H}\bullet \\
& && \mathrm{H}\bullet + \mathrm{Cl_2} \rightarrow \mathrm{HCl} + \mathrm{Cl}\bullet \\
&\text{Termination:} && 2\mathrm{Cl}\bullet \rightarrow \mathrm{Cl_2}
\end{aligned}
.
$$

Hence, our model falls short in the following respects. First, the reaction requires a photochemical initiation where photon absorption is required to cleave the $\mathrm{Cl}-\mathrm{Cl}$ bonds, yet our model initiates reactions spontaneously. Second, the fixed probability p fails to capture the actual Arrhenius behavior of elementary steps. For example, the H-abstraction step $\mathrm{Cl}\bullet + \mathrm{H_2}$ has:

- $E_a^{\text{actual}} = 17$ kJ/mol
- Steric factor ≈ 0.1

compared to our uniform $p = 0.02$. Last, real reactions exhibit chain lengths of 10^4-10^6, meaning each initiation leads to thousands of propagation cycles. Our model treats each collision independently.

8.4. Molecular Simplifications

Note that we only took into acocunt of molecular structures in the steric factor calibrations – even in that occasion the hydrogen and chlorine molecules were simplified as two linearly-connected spheres. However, a starker problem lies within our particle simplifications in later collision simulations. Within the reaction collisions, both hydrogen and chlorine molecules were treated as point particles which are spherically-symmetric and uniform in mass. In reality, however, it would be much better to run the simulation under a "dumbbell" model, where each molecule is accounted by the masses at its ends.

In addition, disregarding free radical collision mechanisms, we also assumed that hydrogen chloride molecules behave as point particles. This assumption elevated our computational prowess, but it failed to account for the deviations caused by asymmetries in the HCl molecule.

9. Conclusion and Final Remarks

Our computational investigation of the $\mathrm{H_2} + \mathrm{Cl_2} \rightarrow 2\mathrm{HCl}$ reaction system successfully bridges microscopic molecular dynamics with macroscopic chemical kinetics, demonstrating the fol-

lowing key findings:

(i) The simulations quantitatively reproduce Maxwell-Boltzmann velocity distributions (Figures 2-9), validating our collision implementation. Temperature-dependent shifts match theoretical predictions.

(ii) Reaction rates exhibit exponential temperature dependence (Figures 9), with extracted activation energy $E_a = 210\text{kJ/mol}$ aligning with literature values (209.2 kJ/mol). The steric factor $s = 0.02$ reflects simplistic but actionable geometric constraints.

Limitations

- We noticed considerable steric factor variations with respect to temperature.
- Concentration to time graphs exhibit non-ideal asymptotic behaviors with fluctuating limiting concentration with respect to temperature.
- The model fails to account for the free radical chain reaction mechanisms.

Strengths

Overall, our work establishes a foundation for predictive computational kinetics in hazardous systems, with methodology generalizable to other gas-phase reactions. The provided implementation grants a testbed for future improvements in molecular simulation fidelity.

Appendix A

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import k, m_p
from numba import jit


num_particles = 1000
mass = 2 * m_p
T = 100
box_size = 1e-8
steps = 5000
dt = 1e-14
radius = 1e-10

#init
np.random.seed(42)
positions = np.random.rand(num_particles, 2) * box_size
velocities = np.random.normal(0, np.sqrt(k * T / mass), (num_particles, 2)
    )

record_every = 100
num_records = steps // record_every
speed_history = np.zeros((num_records, num_particles))

@jit(nopython=True)
def update_system(positions, velocities, speed_history, steps, dt,
    record_every):
     record_index = 0
     for step in range(steps):
         positions += velocities * dt

         # Elastic boundary conditions
         for dim in [0, 1]:
             mask = positions[:, dim] <= 0
             velocities[mask, dim] = np.abs(velocities[mask, dim])
             positions[mask, dim] = 0

             mask = positions[:, dim] >= box_size
             velocities[mask, dim] = -np.abs(velocities[mask, dim])
             positions[mask, dim] = box_size

```

```
        # Record speeds periodically
        if step % record_every == 0:
            speeds = np.sqrt(velocities[:, 0]**2 + velocities[:, 1]**2)
            speed_history[record_index] = speeds
            record_index += 1


update_system(positions.copy(), velocities.copy(), speed_history, steps,
    dt, record_every)
all_speeds = speed_history.flatten()

distribution for 2D
v_theoretical = np.linspace(0, 5000, 200)
pdf_theoretical = (mass / (k * T)) * v_theoretical * np.exp(-mass *
    v_theoretical**2 / (2 * k * T))

plt.figure(figsize=(10, 6))
plt.hist(all_speeds, bins=100, density=True, alpha=0.6, label="Simulated
    Speeds")
plt.plot(v_theoretical, pdf_theoretical, 'r-', label="Theoretical 2D MB")
plt.xlabel("Speed (m/s)")
plt.ylabel("Probability Density")
plt.legend()
plt.grid(True)
plt.show()
```

Listing 1: 2D Molecular Dynamics Simulation

Appendix B

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import k, m_p
from numba import jit, prange

# Constants
mass_H = m_p
mass_Cl = 35.45 * m_p
bond_length_H2 = 0.74e-10
bond_length_Cl2 = 1.99e-10
Ea = 25e3  # J/mol
T = 1000   # K
```

```
box_size = 1e-9
r_crit = 1.0e-10
num_molecules = 500  # H2-Cl2 pairs
steps = 1000
dt = 1e-14
record_interval = 100

# Initialize molecules (vectorized)
@jit(nopython=True)
def initialize_molecules():
    com_H2 = np.random.rand(num_molecules, 2) * box_size
    com_Cl2 = com_H2 + (np.random.rand(num_molecules, 2) - 0.5) * 2e-10

    # Orientations
    theta_H2 = np.random.rand(num_molecules) * 2 * np.pi
    theta_Cl2 = np.random.rand(num_molecules) * 2 * np.pi

    # Atom positions
    H1 = com_H2 + 0.5 * bond_length_H2 * np.column_stack((np.cos(theta_H2)
   , np.sin(theta_H2)))
    H2 = com_H2 - 0.5 * bond_length_H2 * np.column_stack((np.cos(theta_H2)
   , np.sin(theta_H2)))
    Cl1 = com_Cl2 + 0.5 * bond_length_Cl2 * np.column_stack((np.cos(
   theta_Cl2), np.sin(theta_Cl2)))
    Cl2 = com_Cl2 - 0.5 * bond_length_Cl2 * np.column_stack((np.cos(
   theta_Cl2), np.sin(theta_Cl2)))

    # Velocities (Maxwell-Boltzmann)
    v_H2 = np.random.normal(0, np.sqrt(k*T/mass_H), (num_molecules, 2))
    v_Cl2 = np.random.normal(0, np.sqrt(k*T/mass_Cl), (num_molecules, 2))

    return com_H2, com_Cl2, v_H2, v_Cl2, H1, H2, Cl1, Cl2

# Vectorized collision handling
@jit(nopython=True, parallel=True)
def update_collisions(com_H2, com_Cl2, v_H2, v_Cl2):
    for i in prange(num_molecules):
        for j in prange(i+1, num_molecules):
            dx = com_H2[i,0] - com_Cl2[j,0]
            dy = com_H2[i,1] - com_Cl2[j,1]

            # Minimum image convention
            dx -= box_size * np.round(dx/box_size)
```

```
            dy -= box_size * np.round(dy/box_size)

            dist_sq = dx*dx + dy*dy
            if dist_sq < (bond_length_H2 + bond_length_Cl2)**2 /4:
                # Normal vector
                inv_dist = 1.0/np.sqrt(dist_sq)
                nx = dx * inv_dist
                ny = dy * inv_dist

                # Relative velocity
                dvx = v_H2[i,0] - v_Cl2[j,0]
                dvy = v_H2[i,1] - v_Cl2[j,1]

                # Impulse calculation
                J = 2 * (dvx*nx + dvy*ny) / (1/mass_H + 1/mass_Cl)

                # Update velocities
                v_H2[i,0] -= J * nx / mass_H
                v_H2[i,1] -= J * ny / mass_H
                v_Cl2[j,0] += J * nx / mass_Cl
                v_Cl2[j,1] += J * ny / mass_Cl
    return v_H2, v_Cl2

# Fast steric factor calculation
@jit(nopython=True)
def calculate_steric(com_H2, com_Cl2, H1, H2, Cl1, Cl2, v_H2, v_Cl2):
    E_relative = 0.5 * (mass_H*mass_Cl)/(mass_H + mass_Cl) * (
        (v_H2[:,0] - v_Cl2[:,0])**2 +
        (v_H2[:,1] - v_Cl2[:,1])**2 ) * 6.022e23

    steric_count = 0
    energetic_count = 0

    for i in range(num_molecules):
        if E_relative[i] >= Ea:
            energetic_count += 1

            # Check H1 and H2 distances to Cl-Cl midpoint
            Cl_mid = (Cl1[i] + Cl2[i])/2
            d1 = np.sqrt((H1[i,0]-Cl_mid[0])**2 + (H1[i,1]-Cl_mid[1])**2)
            d2 = np.sqrt((H2[i,0]-Cl_mid[0])**2 + (H2[i,1]-Cl_mid[1])**2)

            if d1 < r_crit or d2 < r_crit:
```

```
                steric_count += 1

    return steric_count / max(1, energetic_count)

# Main simulation
com_H2, com_Cl2, v_H2, v_Cl2, H1, H2, Cl1, Cl2 = initialize_molecules()
steric_history = []

for step in range(steps):
    # Update positions
    com_H2 += v_H2 * dt
    com_Cl2 += v_Cl2 * dt

    # Periodic boundaries
    com_H2 %= box_size
    com_Cl2 %= box_size

    # Update atom positions
    H1 = com_H2 + 0.5 * bond_length_H2 * (H1 - com_H2) / np.sqrt(np.sum((H1 - com_H2)**2, axis=1))[:,None]
    H2 = com_H2 - 0.5 * bond_length_H2 * (H2 - com_H2) / np.sqrt(np.sum((H2 - com_H2)**2, axis=1))[:,None]
    Cl1 = com_Cl2 + 0.5 * bond_length_Cl2 * (Cl1 - com_Cl2) / np.sqrt(np.sum((Cl1 - com_Cl2)**2, axis=1))[:,None]
    Cl2 = com_Cl2 - 0.5 * bond_length_Cl2 * (Cl2 - com_Cl2) / np.sqrt(np.sum((Cl2 - com_Cl2)**2, axis=1))[:,None]

    # Handle collisions
    v_H2, v_Cl2 = update_collisions(com_H2, com_Cl2, v_H2, v_Cl2)

    # Record steric factor
    if step % record_interval == 0:
        steric_history.append(calculate_steric(com_H2, com_Cl2, H1, H2, Cl1, Cl2, v_H2, v_Cl2))

# Plot results
plt.figure(figsize=(10,5))
plt.plot(np.arange(len(steric_history))*record_interval*dt, steric_history, 'b-o')
plt.xlabel("Time (s)")
plt.ylabel("Steric factor")
plt.title("Steric Factor Evolution (Optimized Simulation)")
plt.grid(True)
```

```
plt.show()
```

Listing 2: Steric Factor Evolution

Appendix C

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import k, m_p
from numba import jit

# Constants and setup
num_particles = 500
mass_H2 = 2 * m_p
mass_Cl2 = 70 * m_p
box_size = 1e-8
steps = 100  # Keep it short per run to reach t=50
dt = 1e-14
radius = 1e-10
steric_factor = 0.02
record_every = 1  # Record every step
times = np.arange(0, steps * dt, dt)

# Temperature range to investigate
temperature_range = np.linspace(400, 1000, 10)  # Example: 10 points between 400K and 1000K
average_steric_factors = []

@jit(nopython=True)
def simulate(positions, velocities, is_H2, is_Cl2, reacted, steric_factor):
    reaction_counts = np.zeros(steps)
    attempted_reactive_collisions = np.zeros(steps)

    for step in range(steps):
        positions += velocities * dt

        # Wall collisions
        for dim in [0, 1]:
            mask = positions[:, dim] <= 0
            velocities[mask, dim] = np.abs(velocities[mask, dim])
            positions[mask, dim] = 0
```

```
            mask = positions[:, dim] >= box_size
            velocities[mask, dim] = -np.abs(velocities[mask, dim])
            positions[mask, dim] = box_size

        step_reactions = 0
        step_attempts = 0

        for i in range(num_particles):
            if reacted[i]:
                continue
            for j in range(i + 1, num_particles):
                if reacted[j]:
                    continue
                if (is_H2[i] and is_Cl2[j]) or (is_H2[j] and is_Cl2[i]):
                    dx = positions[i, 0] - positions[j, 0]
                    dy = positions[i, 1] - positions[j, 1]
                    dist = (dx**2 + dy**2)**0.5
                    if dist < 2 * radius:
                        step_attempts += 1
                        if np.random.rand() < steric_factor:
                            reacted[i] = True
                            reacted[j] = True
                            step_reactions += 1
                            break

        reaction_counts[step] = step_reactions
        attempted_reactive_collisions[step] = step_attempts

    return reaction_counts, attempted_reactive_collisions

average_steric_factors = []
temperature_range = np.arange(300, 1201, 10)

for T in temperature_range:
    np.random.seed(42)
    positions = np.random.rand(num_particles, 2) * box_size
    masses = np.array([mass_H2] * (num_particles // 2) + [mass_Cl2] * (
   num_particles // 2))
    velocities = np.random.normal(0, 1, (num_particles, 2)) * np.sqrt(k *
   T / masses).reshape(-1, 1)

    is_H2 = np.array([True] * (num_particles // 2) + [False] * (
   num_particles // 2))
```

```
    is_Cl2 = ~is_H2
    reacted = np.zeros(num_particles, dtype=bool)

    reactions, attempts = simulate(positions.copy(), velocities.copy(),
    is_H2, is_Cl2, reacted, steric_factor)

    with np.errstate(divide='ignore', invalid='ignore'):
        steric_factors = np.where(attempts > 0, reactions / attempts, 0.0)

    mean_steric = np.mean(steric_factors[20:51])
    average_steric_factors.append(mean_steric)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(temperature_range, average_steric_factors, marker='o', color='
    teal', label='Avg. steric factor (t=20~50)')
plt.axhline(steric_factor, color='red', linestyle='--', label='Theoretical
    steric factor')
plt.xlabel('Temperature (K)')
plt.ylabel('Average Steric Factor')
plt.title('Temperature Dependence of Average Steric Factor')
plt.grid(True)
plt.legend()
plt.show()
```

Listing 3: Steric Factor over Temperature

Appendix D

```
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations
from scipy.optimize import curve_fit

# Parameters
num_h2 = 100
num_cl2 = 100
box_size = 10.0
sim_time = 100
dt = 0.1
particle_radius = 0.3
p_reaction = 0.02  # Reaction probability upon collision
```

```
temperature = 600  # Kelvin (can be varied)
k_B = 1.380649e-23  # Boltzmann constant (J/K)

# Molecular masses (kg)
mass_h2 = 3.347e-27
mass_cl2 = 1.177e-25
mass_hcl = (mass_h2 + mass_cl2)/2

def maxwell_boltzmann_velocity(mass, temp, num_particles):
    """Generate velocities according to Maxwell-Boltzmann distribution"""
    scale = np.sqrt(k_B * temp / mass)
    return np.random.normal(0, scale, (num_particles, 2))

# Initialize particles with temperature-dependent velocities
particles = []
h2_velocities = maxwell_boltzmann_velocity(mass_h2, temperature, num_h2)
cl2_velocities = maxwell_boltzmann_velocity(mass_cl2, temperature, num_cl2
    )

for i in range(num_h2):
    particles.append({
        'pos': np.random.rand(2) * box_size,
        'vel': h2_velocities[i],
        'type': 'H2',
        'mass': mass_h2
    })
for i in range(num_cl2):
    particles.append({
        'pos': np.random.rand(2) * box_size,
        'vel': cl2_velocities[i],
        'type': 'Cl2',
        'mass': mass_cl2
    })

def handle_collision(p1, p2):
    """Process collision between two particles with conservation of
    momentum"""
    # Check if H2-Cl2 collision
    if {p1['type'], p2['type']} == {'H2', 'Cl2'}:
        if np.random.rand() < p_reaction:
            # Reaction - return 2 new HCl particles with conserved
    momentum
            new_pos = (p1['pos'] + p2['pos']) / 2
```

```
            total_mass = p1['mass'] + p2['mass']
            avg_velocity = (p1['mass']*p1['vel'] + p2['mass']*p2['vel'])/
    total_mass
            # Add thermal fluctuations to product velocities
            thermal_vel = maxwell_boltzmann_velocity(mass_hcl, temperature
    , 2)
            return [
                {'pos': new_pos.copy(),
                 'vel': avg_velocity + thermal_vel[0],
                 'type': 'HCl',
                 'mass': mass_hcl},
                {'pos': new_pos.copy(),
                 'vel': avg_velocity + thermal_vel[1],
                 'type': 'HCl',
                 'mass': mass_hcl}
            ]

    # No reaction - elastic collision with momentum conservation
    r_vec = p1['pos'] - p2['pos']
    r_hat = r_vec/np.linalg.norm(r_vec)
    v_rel = p1['vel'] - p2['vel']
    v_mag = np.dot(v_rel, r_hat)

    # Only collide if approaching each other
    if v_mag < 0:
        j = 2 * p1['mass'] * p2['mass'] * v_mag / (p1['mass'] + p2['mass'
    ])
        p1['vel'] -= j * r_hat / p1['mass']
        p2['vel'] += j * r_hat / p2['mass']

    return [p1, p2]

# Track concentrations
time_points = []
h2_conc = []
cl2_conc = []
hcl_conc = []

# Main simulation loop
for t in np.arange(0, sim_time, dt):
    # Move particles
    for p in particles:
        p['pos'] += p['vel'] * dt
```

```
        # Boundary collisions
        p['pos'] = np.clip(p['pos'], 0, box_size)
        p['vel'] = np.where((p['pos'] <= 0) | (p['pos'] >= box_size), -p['vel'], p['vel'])

    # Detect and process collisions
    new_particles = []
    processed = set()

    for i, p1 in enumerate(particles):
        if i in processed:
            continue

        for j, p2 in enumerate(particles[i+1:], i+1):
            if j in processed:
                continue

            dist = np.linalg.norm(p1['pos'] - p2['pos'])
            if dist < 2 * particle_radius:
                processed.update([i, j])
                new_particles.extend(handle_collision(p1.copy(), p2.copy()))
                break
        else:
            new_particles.append(p1.copy())

    particles = new_particles

    # Record concentrations
    counts = {'H2':0, 'Cl2':0, 'HCl':0}
    for p in particles:
        counts[p['type']] += 1

    time_points.append(t)
    h2_conc.append(counts['H2'] / (num_h2 + num_cl2))
    cl2_conc.append(counts['Cl2'] / (num_h2 + num_cl2))
    hcl_conc.append(counts['HCl'] / (num_h2 + num_cl2))

# Define fitting functions
def exp_decay(t, a, k, c):
    return a * np.exp(-k * t) + c

def exp_growth(t, a, k, c):
```

```
    return a * (1 - np.exp(-k * t)) + c

# Fit curves
p0_h2 = [1, 0.1, 0]  # Initial guesses for amplitude, rate, offset
popt_h2, pcov_h2 = curve_fit(exp_decay, time_points, h2_conc, p0=p0_h2)
popt_cl2, pcov_cl2 = curve_fit(exp_decay, time_points, cl2_conc, p0=p0_h2)
popt_hcl, pcov_hcl = curve_fit(exp_growth, time_points, hcl_conc, p0=p0_h2
    )

# Generate fitted curves
fit_h2 = exp_decay(np.array(time_points), *popt_h2)
fit_cl2 = exp_decay(np.array(time_points), *popt_cl2)
fit_hcl = exp_growth(np.array(time_points), *popt_hcl)

# Plot results
plt.figure(figsize=(10, 6))

# Plot raw data
plt.plot(time_points, h2_conc, 'b-', label='[ H  ] (sim)', alpha=0.3)
plt.plot(time_points, cl2_conc, 'g-', label='[ C l  ] (sim)', alpha=0.3)
plt.plot(time_points, hcl_conc, 'r-', label='[HCl] (sim)', alpha=0.3)

# Plot fitted curves
plt.plot(time_points, fit_h2, 'b--', label=f'Fit  H  : {popt_h2[0]:.2f}exp
    (-{popt_h2[1]:.2f}t) + {popt_h2[2]:.2f}')
plt.plot(time_points, fit_cl2, 'g--', label=f'Fit  C l  : {popt_cl2[0]:.2f}
    exp(-{popt_cl2[1]:.2f}t) + {popt_cl2[2]:.2f}')
plt.plot(time_points, fit_hcl, 'r--', label=f'Fit HCl: {popt_hcl[0]:.2f
    }(1-exp(-{popt_hcl[1]:.2f}t)) + {popt_hcl[2]:.2f}')

plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title(f' H   +  C l      2HCl Reaction Kinetics\n(T={temperature}K,
    p_reaction={p_reaction})')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Listing 4: Steric Factor over Temperature

Bibliography

Suresh Sharma. H2 – y2 reactions (y = cl, br, i): A comparative and mechanistic aspect. *Journal of Chemistry, Environmental Sciences and its Applications*, 1:45–51, 09 2014. doi: 10.15415/jce.2014.11005.