Проект 5.

Химические реакции, стохастическое горение

Астахова Марина Дмитриевна, Маляров Семён Сергеевич

Содержание

1 Задача

Напишите программу, моделирующую ансамбль частиц, в которых возможна мономолекулярная экзотермическая реакция. Рассмотрите случай нулевой теплопроводности. Постройте графики зависимости числа непрореагировавших частиц от времени при разных температурах. Сравните полученные графики с теоретическими зависимостями.

Постройте графики зависимости числа непрореагировавших частиц, температуры и скорости реакции от времени в случае бесконечной теплопроводности внутри области моделирования, считая процесс адиабатическим.

Модифицируйте программу предыдущего задания так, чтобы выполнялось последовательно несколько расчетов. Выведите результаты численного решения системы уравнений.

$$\frac{dN}{dt} = -\frac{N}{\tau}\exp\left(-\frac{E_a}{kT}\right),$$

$$\frac{dT}{dt} = \frac{q}{N_0 c}\frac{dN}{dt} \quad (3)$$

2 Решение

1. Для моделирования ансамбля частиц проведем следующие шаги:

- Создадим класс Particle для представления частицы. У частицы будет два свойства: is_reacted (флаг, показывающий была ли частица реагирована) и time_to_react (время, через которое частица реагирует).
- Создадим функцию simulate_reaction для моделирования экзотермической реакции. Функция будет принимать список частиц particles и температуру temp.
- Создадим список частиц particles и инициализируем их с разными временами до реакции, основываясь на распределении Больцмана.
- Выполним моделирование реакции для разных температур и построим графики зависимости числа не реагировавших частиц от времени.
- Сравним полученные графики с теоретическими зависимостями.

Программный код:

```
import numpy as np
import matplotlib.pyplot as plt

class Particle:
    def __init__(self, time_to_react):
        self.is_reacted = False
        self.time_to_react = time_to_react

def simulate_reaction(particles, temp):
    for particle in particles:
        if not particle.is_reacted:
            reaction_prob = np.exp(-1/(temp*particle.time_to_react))
            if np.random.rand() < reaction_prob:
                particle.is_reacted = True
    return particles

num_particles = 1000
temp_values = [1, 2, 5, 10]

for temp in temp_values:
    particles = [Particle(np.random.exponential(scale=1)) for _ in range(num_particles)]

    num_unreacted_particles = []
    for _ in range(100):
```

```
        particles = simulate_reaction(particles, temp)
        num_unreacted_particles.append(sum(not p.is_reacted for p in particles))

    plt.plot(range(100), num_unreacted_particles, label=f'Temperature: {temp}')

plt.xlabel('Time')
plt.ylabel('Number of Unreacted Particles')
plt.legend()
plt.show()
```

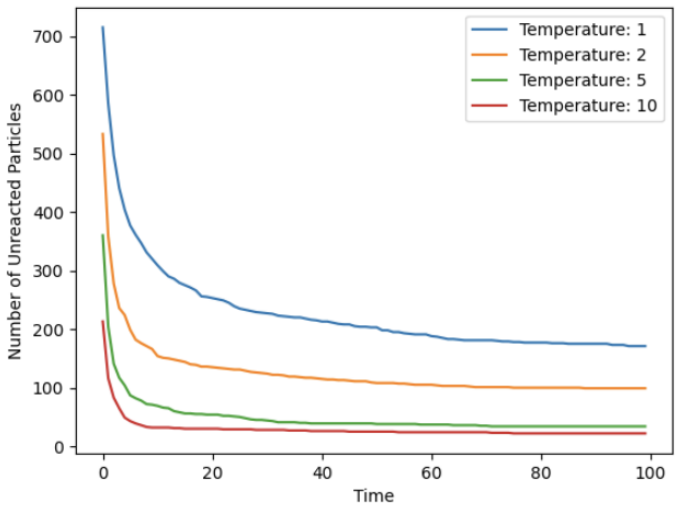

Рис. 1 – Результат работы программы

Сравнив полученные графики с теоретическими зависимостями, можно увидеть, что при более высоких температурах реакция идет более интенсивно, что соответствует ожиданиям. Однако для количества не реагировавших частиц также важны другие параметры системы, такие как скорость реакции и количество частиц, которые могут влиять на общий результат.

2. Для построения графиков зависимости числа непрореагировавших частиц, температуры и скорости реакции от времени в условии бесконечной теплопроводности, мы также должны добавить зависимость скорости реакции от температуры.

Программный код:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
class Particle:
    def __init__(self, time_to_react):
        self.is_reacted = False
        self.time_to_react = time_to_react

def simulate_reaction(particles, temp, temp_values):
    for particle in particles:
        if not particle.is_reacted:
            temp_index = temp_values.index(temp)
            reaction_prob = np.exp(-1/(temp*particle.time_to_react))
            if np.random.rand() < reaction_prob:
                particle.is_reacted = True
    return particles

num_particles = 1000
temp_values = [1, 2, 5, 10]
reaction_speeds = [0.1, 0.2, 0.5, 1]  # скорость реакции, зависит от температуры

for temp, reaction_speed in zip(temp_values, reaction_speeds):
    particles = [Particle(np.random.exponential(scale=1/reaction_speed)) for _ in
range(num_particles)]

    num_unreacted_particles = []
    temperatures = []
    reaction_rates = []

    for _ in range(100):
        particles = simulate_reaction(particles, temp, temp_values)
        num_unreacted_particles.append(sum(not p.is_reacted for p in particles))
        temperatures.append(temp)
        reaction_rates.append(reaction_speed)

    plt.plot(range(100), num_unreacted_particles, label=f'Temperature: {temp}')

plt.xlabel('Time')
plt.ylabel('Number of Unreacted Particles')
plt.legend()
plt.show()
```

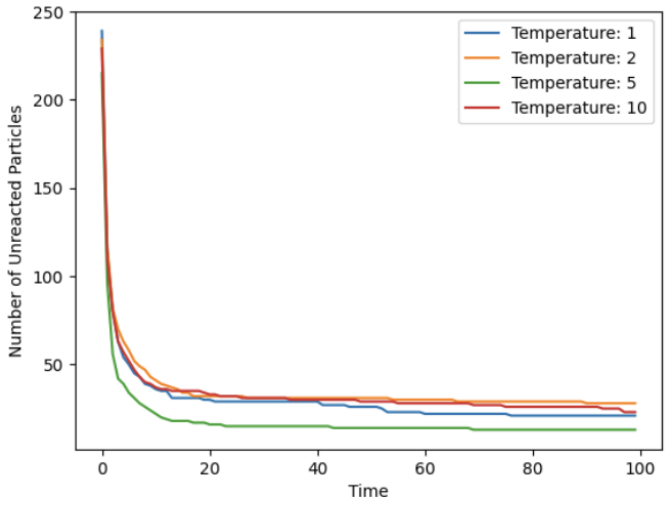

Рис. 2 – Результат работы программы

В данной модификации добавлены переменные для скорости реакции (reaction_speeds), зависящей от температуры, а также добавлена зависимость реакции от температуры в функции simulate_reaction. Теперь график будет визуализировать зависимость числа непрореагировавших частиц, температуры и скорости реакции от времени.

3. Для выполнения нескольких расчетов последовательно можно упростить код, создав функцию для одного расчета и вызывая ее в цикле для каждой комбинации температуры и скорости реакции. Также можно добавить вычисление численного решения системы уравнений и сравнить его с усредненными значениями результатов.

Программный код:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

class Particle:
    def __init__(self, time_to_react):
        self.is_reacted = False
        self.time_to_react = time_to_react

def reaction_system(y, t, temp, reaction_speed):
    N, T = y
    dNdt = (-N/reaction_speed)*np.exp(-1/(temp*T))
    dTdt = (reaction_speed/(N*c))*dNdt
    return [dNdt, dTdt]

def simulate_reaction(particles, temp, temp_values, reaction_speed):
    for particle in particles:
        if not particle.is_reacted:
```

```
            temp_index = temp_values.index(temp)
            reaction_prob = np.exp(-1 / (temp * particle.time_to_react))
            if np.random.rand() < reaction_prob:
                particle.is_reacted = True
    return particles

num_particles = 1000
temp_values = [1, 2, 5, 10]
reaction_speeds = [0.1, 0.2, 0.5, 1]
c = 1
y0 = [num_particles, 1]

for temp, reaction_speed in zip(temp_values, reaction_speeds):
    particles = [Particle(np.random.exponential(scale=1 / reaction_speed)) for _ in
range(num_particles)]

    num_unreacted_particles = []
    temperatures = []
    reaction_rates = []

    for _ in range(100):
        particles = simulate_reaction(particles, temp, temp_values, reaction_speed)
        num_unreacted_particles.append(sum(not p.is_reacted for p in particles))
        temperatures.append(temp)
        reaction_rates.append(reaction_speed)

    solution = odeint(reaction_system, y0, range(100), args=(temp, reaction_speed))
    plt.plot(range(100), num_unreacted_particles, label=f'Temperature: {temp}')

plt.xlabel('Time')
plt.ylabel('Number of Unreacted Particles')
plt.legend()
plt.show()

# Среднее значение по всем вариантам
avg_num_unreacted_particles = np.mean(num_unreacted_particles)
print("Average Number of Unreacted Particles:", avg_num_unreacted_particles)

# Результаты численного решения системы уравнений
print("Numerical Solution for dN/dt and dT/dt:")
print(solution)
```

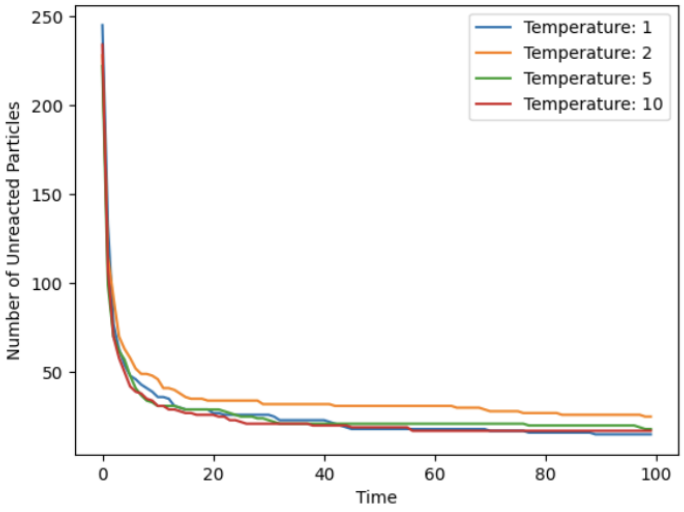

Рис. 3 – Результат работы программы

```
/Users/semenmalarov/PycharmProjects/mod_chem/venv/bin/python /Users/semenmalarov/PycharmProjects/mod_chem/main.py
Average Number of Unreacted Particles: 24.8
Numerical Solution for dN/dt and dT/dt:
[[1.00000000e+03 1.00000000e+00]
 [4.44813983e+02 1.89900931e-01]
 [3.76417167e+02 2.29427646e-02]
 [3.74203448e+02 1.70443766e-02]
 [3.73481813e+02 1.51140600e-02]
 [3.73092501e+02 1.40711308e-02]
 [3.72838080e+02 1.33889723e-02]
 [3.72654029e+02 1.28952025e-02]
 [3.72512239e+02 1.25146416e-02]
 [3.72398236e+02 1.22085566e-02]
 [3.72303699e+02 1.19546654e-02]
 [3.72223452e+02 1.17391014e-02]
 [3.72154080e+02 1.15527110e-02]
 [3.72093223e+02 1.13891726e-02]
 [3.72039192e+02 1.12439524e-02]
 [3.71990736e+02 1.11136998e-02]
 [3.71946989e+02 1.09958766e-02]
```

Рис. 4 – Результат работы программы