Thermalize

May 23, 2023

1 Simulation of Thermalization in an Oscillating Particle: A Thermal Physics Project

Alfaifi, Ammar – 201855360

Thermalization is a fundamental concept in thermal physics, describing the process by which a system reaches thermal equilibrium with its surroundings. Understanding and studying thermalization is crucial for various scientific fields, including condensed matter physics, statistical mechanics, and quantum mechanics.

In this thermal physics project, we focus on simulating the thermalization of an oscillating particle. The project aims to investigate how an oscillating particle interacts with a heat reservoir and eventually reaches a state of thermal equilibrium.

The simulation utilizes numerical methods and Monte Carlo techniques to model the energy exchange between the oscillating particle and the heat reservoir. By incorporating principles from statistical mechanics, we can gain insights into the relaxation dynamics and energy distribution of the particle as it undergoes thermalization.

The project involves several key steps. Initially, we define the parameters, such as the number of particles in the reservoir, the temperature of the heat bath, and the amplitude of the particle's oscillation. These parameters play crucial roles in shaping the thermalization process.

Through a series of computational iterations, we simulate the interactions between the oscillating particle and the reservoir. At each time step, energy exchanges occur, influenced by random fluctuations, allowing the particle to reach a state of thermal equilibrium. We monitor the evolution of the energy distribution and observe how it converges towards the distribution dictated by the heat reservoir's temperature.

To visualize and analyze the thermalization process, we create an animation that showcases the changing energy distribution over time. By observing the animation and analyzing statistical properties, we gain a deeper understanding of the mechanisms underlying thermalization and the

1.1 Setup the Python packages

[21]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import constants as con
from matplotlib_inline.backend_inline import set_matplotlib_formats
```

```
set_matplotlib_formats('pdf')
plt.rcParams |= {
    'text.usetex': True,
    'figure.figsize': (10, 4)
}
sns.set_theme()
set_matplotlib_formats('svg', 'pdf')
```

Assume two particle particle one has a mass of m_1, and the other with much smmaller mass of m_2

[45]:
```
m2 = 1.00784 * con.u # He mass
m1 = 10 * m2
omega = 2 * np.pi * 0.5
k = omega**2 * m1

def v1(t, amp=1):
    return - amp * omega * np.sin(omega * np.random.rand())

def v2_after(v1, v2):
    """Final velocity of particle 2 after collision
    with particle 1."""
    return 2*m1 / (m1+m2) * v1 - (m1-m2) / (m1+m2) * v2

def temp(v):
    return m2 * v**2 / (3 * con.k)
```

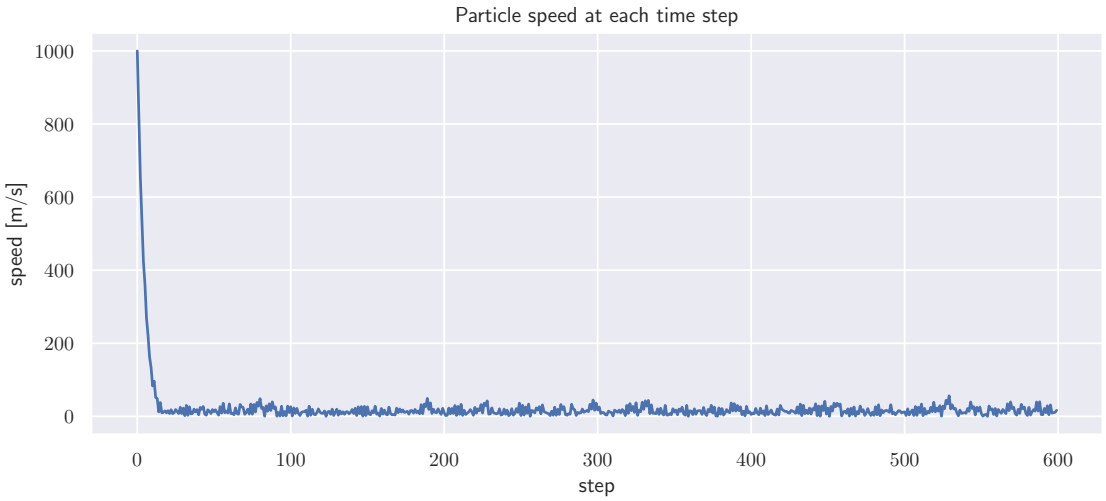
1.2 Demostrating Thermalization

1.2.1 The Stablization of Speed

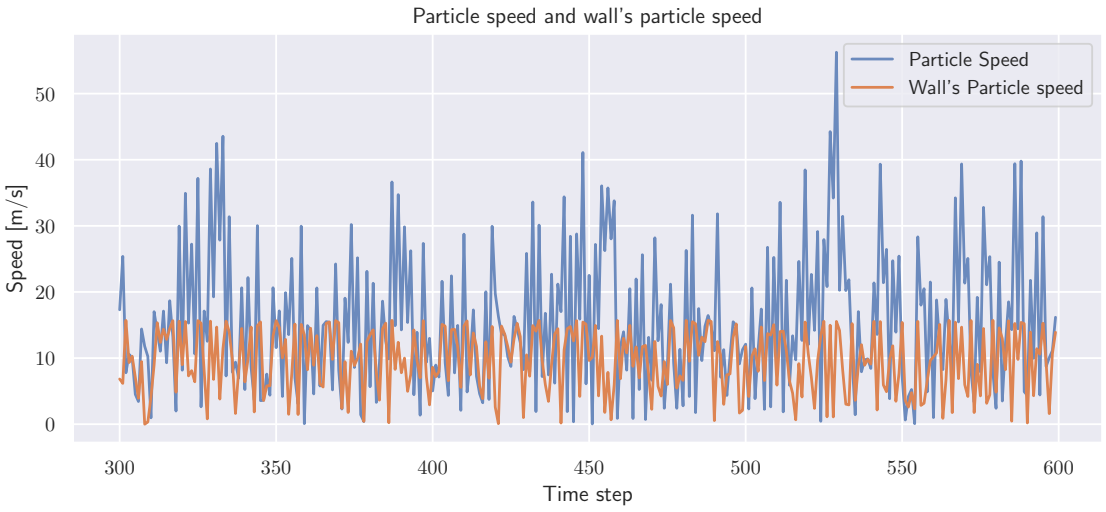
[49]:
```
steps = 600
A = 5
velocities = np.zeros(steps)
osc_velocities = np.zeros(steps)
time = np.arange(steps)
# initial speed
velocities[0] = 1000

for i in time[1:]:
    osc_velocities[i] = v1(i, A)
    velocities[i] = v2_after(osc_velocities[i], velocities[i-1])

avg_speed = abs(velocities[300:]**2).sum() / len(velocities[300:])
print(f'Avg energy is {m2 * velocities[300:].var() / con.e} eV')
print(f'Total energy is {0.5* k*A**2 / con.e} eV')
```

```
plt.plot(time, abs(velocities))
plt.title("Particle speed at each time step")
plt.xlabel('step')
plt.ylabel('speed [m/s]')
plt.show()
```

```
Avg energy is 2.7161644548245293e-06 eV
Total energy is 1.2886650588822158e-05 eV
```



[50]:
```
plt.plot(time[300:], abs(velocities[300:]), label='Particle Speed', alpha=0.8)
plt.plot(time[300:], abs(osc_velocities[300:]), label="Wall's Particle speed")
plt.title("Particle speed and wall's particle speed")
plt.xlabel("Time step")
plt.ylabel("Speed [m/s]")
plt.legend()
plt.show()
```



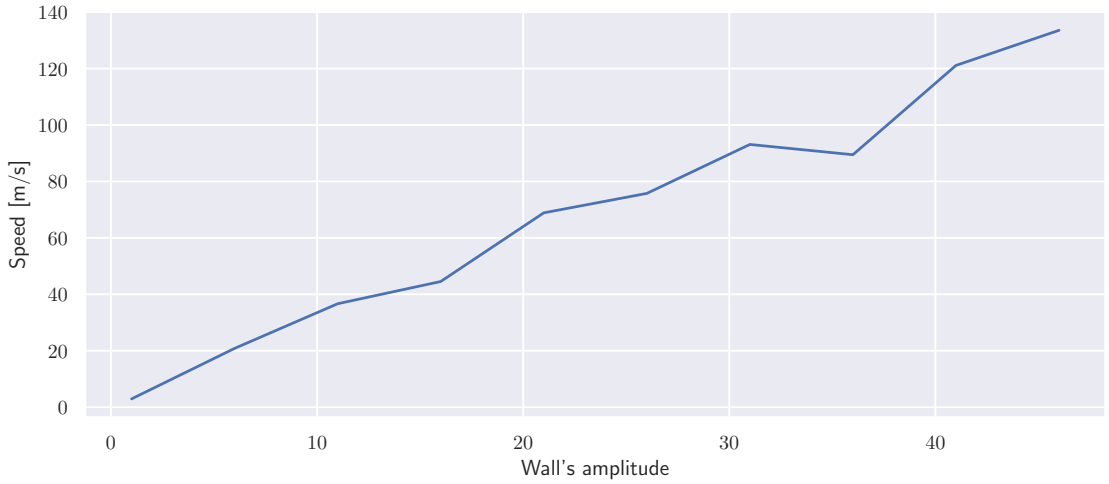
1.2.2 Reaching Equilibrium Temperature

[61]:
```
print(f"Temperature of particle is {m2 * velocities[300:].var():0.3}, and temp␣
↪of wall {0.5* k*A**2:.3}")
```

```
Temperature of particle is 4.35e-25, and temp of wall 2.06e-24
```

1.2.3 Equality of Speed

[63]:
```
amps = np.arange(1, 50, 5)
variances = []
for amp in amps:
    for i in time[1:]:
        velocities[i] = v2_after(v1(i, amp), velocities[i-1])
    variances.append(velocities[300:].var())

plt.plot(amps, [np.sqrt(x) for x in variances])
plt.ylabel("Speed [m/s]")
plt.xlabel("Wall's amplitude")
plt.show()
```


140
120
100
80
60
40
20
0
Speed [m/s]
0
10
20
30
40
Wall's amplitude

[ ]: