布朗运动法确定阿伏伽德罗常数：实验报告

刘泓尊　2018011446　计 84

liu-hz18@mails.tsinghua.edu.cn

2020 年 5 月 16 日

目录

1 实验目的

a. 通过对布朗运动的观测和数据统计分析，得出扩散系数，确定阿伏伽德罗常数.

2 实验原理

2.1 布朗运动

布朗运动是指悬浮在液体或气体中的微粒所做的不停息的无规则运动，由苏格兰植物学家布朗 1827 年发现花粉在水中的运动而得名。这个现象揭示了液体、气体等物质是由大量原子或分子构成的事实。

2.2 郎之万方程

法国物理学家郎之万提出了描述布朗运动的数学方程:

$$m\frac{dv}{dt} = -\gamma v + \xi$$

其中 m 是微粒的质量，v 是微粒的速度，ξ 是噪声，具有随机性，是微粒不规则运动的来源，其系综平均值为 0，不同时刻的噪声之间没有关联，即 $\langle \xi(t_i)\xi(t_j)\rangle = A\delta(t_i - t_j)$；$-\gamma v$ 是微粒所受的阻力，是微粒所受力的系综平均值。

布朗运动是一个二维平面的运动，两个维度的运动互不关联，因此可以看作是微粒在 X 方向的运动方程，Y 方向的运动方程形式相同。

假设微粒为球形，微粒直径为 d，流体粘度为 η，根据 Stokes 公式得到

$$\gamma = 3\pi d\eta$$

由于阻力项和噪声项都来自于流体原子分子施加给微粒的合力，所以 γ 和 A 之间存在联系，这种联系由涨落耗散定理决定，即

$$A = 2k_B T\gamma$$

其中 T 是流体的温度，k_B 是玻尔兹曼常数。

2.3 郎之万方程的几个结论

求解郎之万方程，当观测时刻远大于微粒趋向于热平衡的弛豫时间时，有 (1) 微粒平均动能的 X 分量 (能均分定理)

$$\frac{1}{2}m\langle v^2\rangle = \frac{1}{2}k_B T$$

(2) 微粒 X 方向位移的平均平方偏差

$$\langle [x(t) - x(t_0)]^2\rangle = 2Dt$$

其中 D 是扩散系数, 有爱因斯坦关系决定

$$D = \frac{k_B T}{\gamma}$$

我们还可以得到 (3) 微粒每隔时间 τ 的位移平方平均值, 在 τ 足够大时

$$\langle (\triangle x)^2\rangle = 2D\tau$$

根据上述三个关系，我么可以通过观测数据得到 D 值，求出玻尔兹曼常数 k_B，再根据气体常数 $R = N_A k_B$ 确定阿伏伽德罗常数 N_A.

3 实验方法

微粒布朗运动的实验观测需要使用超显微镜，要求较高。我采用计算机数值计算的方法求解郎之万方程，代替真实实验观测。参数如下：

1. 微粒直径: $d = 1\mu m$
2. 微粒质量: $m = 1\mu g$
3. 流体温度: $T = 293K$
4. 流体粘度: $\eta = 0.1cP = 10^{-4} Pa \cdot s$
5. 初始观测时刻: $t_0 = 0s$
6. 微粒初始坐标: $x_0 = 0\mu m$
7. 微粒初始速度: $v_0 = 0\mu m/s$

运行程序将给出时间间隔为 $\tau = 30s$ 的一系列时刻微粒的坐标 x 和位移 $\triangle x$，相当于一次连续观测的结果。计算误差相当于测量误差。

4 数据处理

首先计算阻力系数

$$\gamma = 3\pi d\eta = 9.4248 \times 10^{-10}(SI)$$

4.1 布朗运动轨迹

运行程序两次，将输出坐标分别作为微粒的 X 和 Y 方向的位置随时间的变化，得到的微粒轨迹如下:

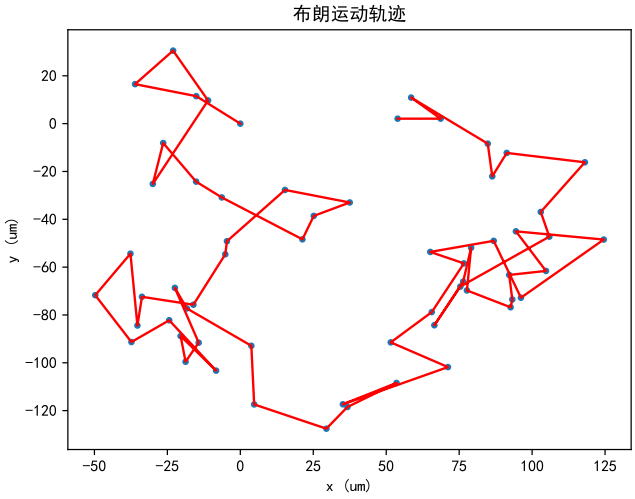
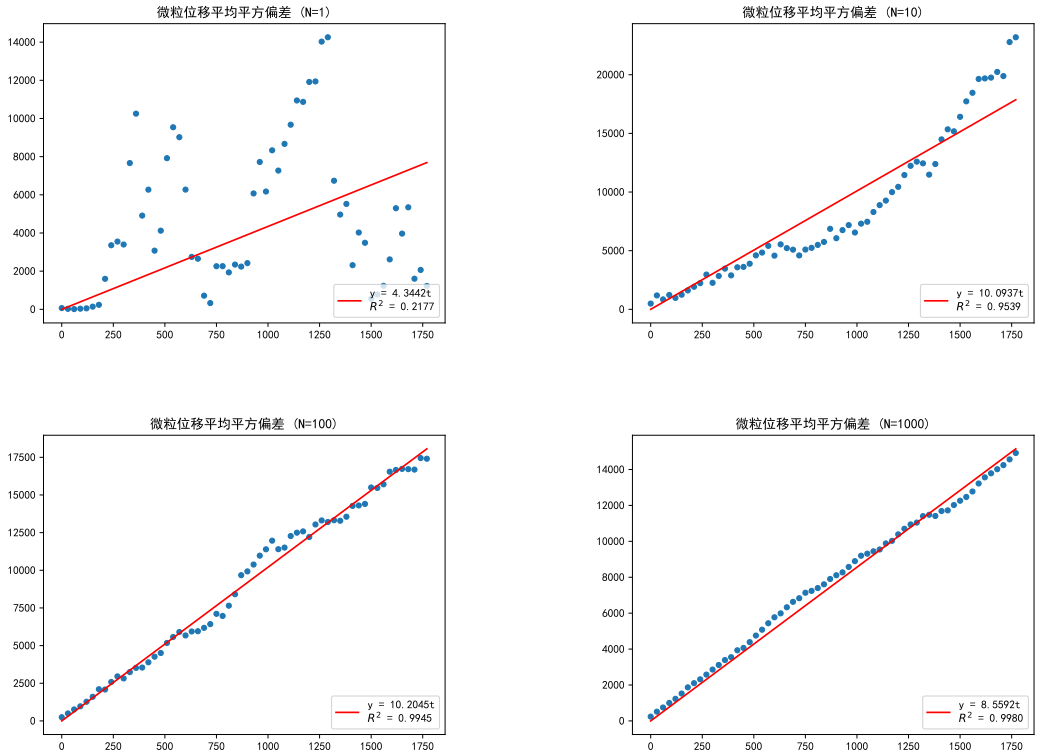


图 1: 二维布朗运动轨迹图

4.2 微粒位移平均平方偏差 $\langle x^2(t)\rangle$

按实验方法统计微粒位移的平均平方偏差 $\langle x^2(t)\rangle (N = 1, 10, 100, 1000)$. 进行数据拟合后得到的图像如下图所示，相应的拟合参数和相关系数在图中给出。



(a) (b)

图 2: 微粒位移平均平方偏差

从 $N = 1000$ 的数据中可以看到 $R^2 = 0.9980$，线性度已经满足要求。过零线性拟合后得到 $y = 8.5592t$, 利用公式 (2) 可以得到

$$D = \frac{8.5592}{2} = 4.280 \times 10^{-12} m^2/s$$

根据公式 $D = \frac{k_B T}{\gamma}$ 得到玻尔兹曼常数

$$k_B = \frac{\gamma D}{T} = 1.377 \times 10^{-23} J/K$$

进一步得到阿伏伽德罗常数

$$N_A = \frac{R}{k_B} = \frac{8.314}{1.377 \times 10^{-23}} = 6.037 \times 10^{23} mol^{-1}$$

和标准值的相对误差:

$$\epsilon = \frac{|6.037 - 6.022|}{6.022} \times 100\% = 0.26\%$$

可以看到和标准值符合的很好。

4.3 dt 内微粒位移平方平均值 $\langle(\triangle x)^2\rangle$

统计观测间隔 τ 内微粒位移平方平均值 $\langle(\triangle x)^2\rangle$(N=100, 1000) 得到的结果如下:

N	$\langle(\triangle x)^2\rangle\ (\mu m^2)$
100	254.213
1000	248.882

利用 $N = 1000$ 的结果得到 D:

$$D = \frac{\langle(\triangle x)^2\rangle}{2dt} = 4.148 \times 10^{-12} m^2/s$$

根据公式 $D = \frac{k_B T}{\gamma}$ 得到玻尔兹曼常数

$$k_B = \frac{\gamma D}{T} = 1.334 \times 10^{-23} J/K$$

进一步得到阿伏伽德罗常数

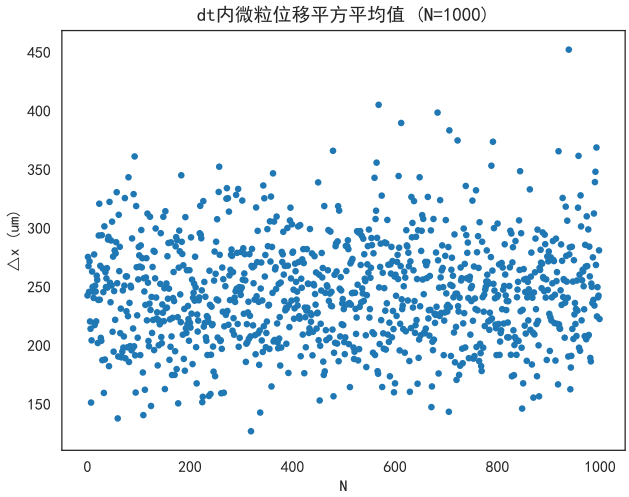
$$N_A = \frac{R}{k_B} = \frac{8.314}{1.334 \times 10^{-23}} = 6.231 \times 10^{23} mol^{-1}$$

和标准值的相对误差:

$$\epsilon = \frac{|6.231 - 6.022|}{6.022} \times 100\% = 3.47\%$$

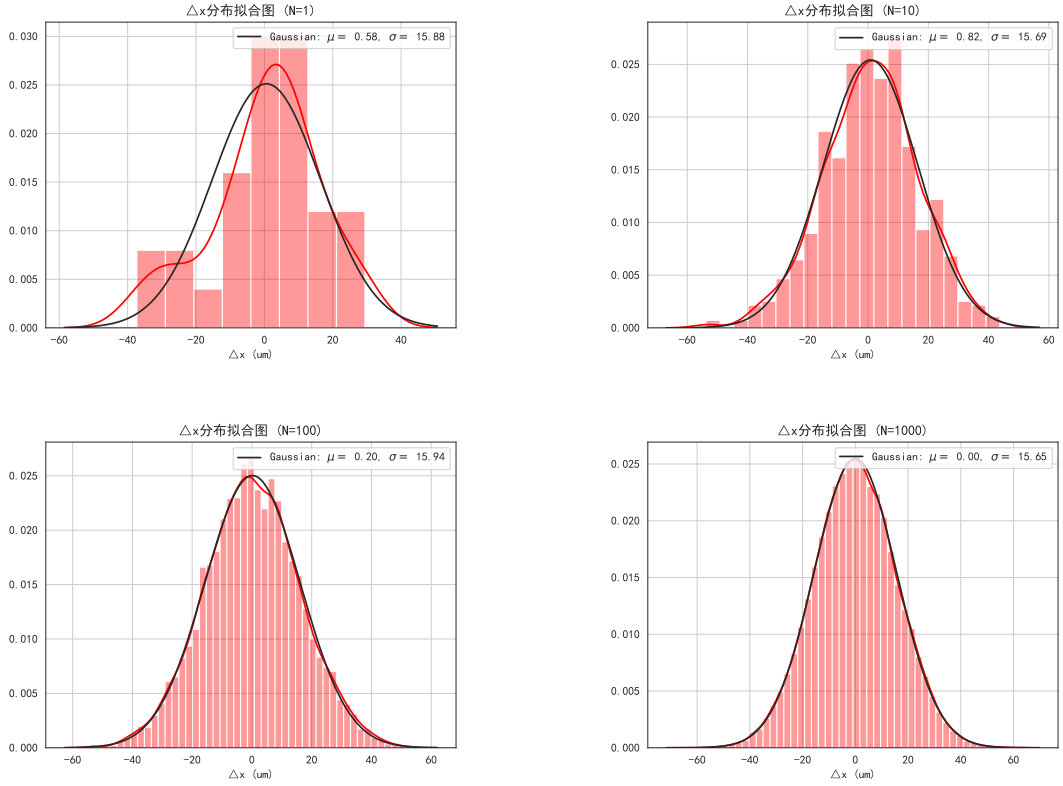
和标准值有细微的偏差，初步判断是 dt 取的还不够充分长。

$N = 1000$ 时的数据 $\triangle x^2$ 散点图如下:



4.4 $\triangle x$ 的分布

对 $N = 1, 10, 100, 1000$ 进行直方图统计，可以看到数据呈现正态分布。进行正态分布拟合得到的参数 (μ, σ) 在图中标出。



(a) (b)

图 3: 微粒位移平均平方偏差

在 $N = 1000$ 时，得到分布

$$\triangle x \sim N(0.00, 15.65)$$

5 讨论题

5.1 什么是流体中原子或分子的平均自由程？确定阿伏伽德罗常数后如何估计原子或分子大小？

分子在运动过程中不断碰撞，由于分子间作用力产生排斥。我们可以将分子看做弹性球，分子在两次碰撞之间保持匀速直线运动，在这段时间内分子所经过的路程称作自由程，记为 λ. 由于每两次碰撞之间的状态不尽相同，这一物理量在宏观上使用统计意义，即取其平均值，称为平均自由程 (Mean Free Path), 记做 $\overline{\lambda}$. 如果设分子数密度为 n, 分子直径为 d, 则 $\overline{\lambda}$ 的公式为:

$$\overline{\lambda} = \frac{1}{\sqrt{2} n \pi d^2}$$

液体和固体往往采用分子密堆积构型，分子之间紧密排列，并且可以视作球形，分子体积为

$$V = \frac{4\pi}{3} \left(\frac{d}{2} \right)^3$$

设固体或液体的密度为 ρ, 摩尔质量为 M, 在得到阿伏伽德罗常数之后，可以计算分子体积:

$$V = \frac{M}{\rho N_A}$$

由上述两式可以得到:

$$d = \sqrt[3]{\frac{6M}{\pi \rho N_A}}$$

5.2 数值计算所使用的参数是否合理 (微粒的质量、直径; 流体的温度、粘度)? 如果不合理对结果有何影响?

经过查阅文献，我找到了皮兰进行实验的数据:

$$d = 7.34 \times 10^{-7} m$$

$$\eta = 1.14 \times 10^{-3} Pa \cdot s$$

我使用上述数据进行测试，统计观测间隔 τ 内微粒位移平方平均值 $\langle(\triangle x)^2\rangle$(N=1000) 采用 4.3 所用方法得到:

$$D = 0.514$$

进而得到

$$k_B = \frac{3\pi d\eta D}{T} = 1.385 \times 10^{-23} J/K$$

$$N_A = \frac{R}{k_B} = 6.004 \times 10^{23}$$

相对误差

$$\epsilon = \frac{|6.004 - 6.022|}{6.022} \times 100\% = 0.30\%$$

可以看到误差减少了很多!

通过上面的测试可以初步得到结论: 液体粘度的适当增加、液滴尺度的适当减少将使得实验结果更为精确。液滴尺度减小时，其运动受自身重力的影响越小，更加贴近布朗运动。当然，上述结论都建立在数值模拟与实际现象吻合的情况，我没有条件验证数值程序是否合理，只能根据模拟结果做些许推测。其中奥妙万望老师教诲!

6 实验小结

本次实验我学习了布朗运动与郎之万方程的相关知识，体会到了数值计算在模拟物理问题中的极大便利。通过对模拟结果的拟合、统计，我加深了对布朗运动的了解，尤其是当我画出 $\triangle x$ 的分布时，粒子分布与正态分布达到了很完美的契合程度，这让我再次领略到了物理之美。

经过数值模拟与统计分析，我得到了较好的实验结果。十分感谢老师的悉心指导!

A 批量测试及绘图程序

为了便于批量运行程序以及绘制统计、拟合图像，我编写了 python 脚本来运行程序并统计结果。具体代码如下:

main.py

```python
# 布朗运动批量测试脚本
# CopyRight @ LiuHongzun at THU.CST 2020.5
import os, time, math
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize, stats
from scipy.stats import norm # 用于拟合正态分布曲线
import seaborn as sns

T = 293.0 # K
gamma = 3.0 * math.pi * 0.1 # 3pi * 0.0001 Pa.s * 10^-6 m = 3pi*10^-10
R = 8.314 # J.mol.K^-1
tau = 30

def compile():
    if 0 != os.system('g++ brown.cpp -o brown.exe'):
        print('compile failed!')

def run(times):
    for t in range(times):
        if 0 != os.system('brown > results/out_%d.txt' % (t)):
            print('run batch %d failed' % (t))
        print('running program id:%d' % t, end='\r')
        time.sleep(2)

def linear(x, A):
    return A * x

def static_1(n):
    mat = np.zeros((n, 60))
    for t in range(n):
        with open('results/out_%d.txt' % (t), 'r', errors='ignore') as f:
            for i, line in enumerate(f.readlines()[2:]):
                mat[t][i] = float(line.strip().split()[1])
    array = np.mean(mat**2, axis=0)
    mean_array = np.arange(0, 1800, 30)
    # print(mean_array)
    A = optimize.curve_fit(linear, mean_array, array)[0][0]
    D = A / 2
```

```
    k_b = gamma * D / T
    N_a = R / k_b / 100.0
    print('A', A, 'D', D, 'k_b', k_b, 'NA', N_a)
    plt.figure()
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.scatter(mean_array, array, s=20)
    pred = A * mean_array
    plt.plot(mean_array, pred, color='red')
    plt.legend(['y = {:.4f}t'.format(A)], loc='lower right')
    plt.title('微粒位移平均平方偏差 (N=%d)' % (n))
    plt.savefig('curve_%d.pdf' % (n), dpi=300)


def plot_distribution(distribution, n):
    sns.set_style('white')
    plt.figure()
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.grid(True)
    sns.distplot(distribution, fit=norm, color='red')
    (mu, sigma) = norm.fit(distribution)
    plt.legend(['Gaussian: $\mu=$ {:.2f}, $\sigma=$ {:.2f}'. \
    format(mu, sigma)], loc='upper right')
    plt.title('x分布拟合图 (N=%d)' % n)
    plt.xlabel('x (um)')
    plt.savefig('dis_%d.pdf' % (n), dpi=300)


def static_2(n):
    mat = np.zeros((n, 60))
    for t in range(n):
        with open('results/out_%d.txt' % (t), 'r', errors='ignore') as f:
            for i, line in enumerate(f.readlines()[2:]):
                mat[t][i] = float(line.strip().split()[2])
    distribution = mat.flatten()
    plot_distribution(distribution, n)
    array = np.mean(mat**2, axis=1)
    delta_x_square = np.mean(array) # 10^-12 m
    D = delta_x_square / (2*tau) # 10^23
    k_b = gamma * D / T
    N_a = R / k_b
    print('delta_x^2', delta_x_square, 'D', D, 'k_b', k_b, 'N_A:', N_a/100)
    mean_array = np.arange(0, n, 1)
```

```
    plt.figure()
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title('dt内微粒位移平方平均值 (N=%d)' % (n))
    plt.xlabel('N')
    plt.ylabel('⍰x (um)')
    plt.scatter(mean_array, array, s=10)
    # print(mean_array, array)
    plt.savefig('curve2_%d.pdf' % (n), dpi=300)


def plot():
    # read x, y
    x, y = [], []
    with open('results/out_0.txt', 'r', errors='ignore') as f:
        for line in f.readlines()[1:]:
            x.append(float(line.strip().split()[1]))
    with open('results/out_500.txt', 'r', errors='ignore') as f:
        for line in f.readlines()[1:]:
            y.append(float(line.strip().split()[1]))
    # print(x, y)
    plt.figure()
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title(r'布朗运动轨迹')
    plt.xlabel('x (um)')
    plt.ylabel('y (um)')
    plt.plot(x, y, color='red')
    plt.scatter(x, y, s=10)
    plt.savefig('plot_brown.pdf', dpi=300)


def main():
    os.makedirs('results', exist_ok=True)
    t = 10
    compile()
    run(times=t)
    plot()
    static_1(n=t)
    static_2(n=t)

if __name__ == '__main__':
    main()
```