**《非平衡统计物理专题》第一次作业**

（作业提交截止时间： 3月22日21:00）

说明：

1）文件名格式范例：“第一次作业\_19720192203306\_程##.docx”

1. 有粒子处在简谐势阱中,其哈密顿表示如下：



现放置该粒子于温度为的热浴中作布朗运动。

1）试写出刻画该粒子运动的郎之万方程；

2）解析计算该粒子能量平均值随温度的变化规律；

3）采用分子动力学模拟（随机RK2算法）验算2）的结果。

解：

1）该粒子运动的郎之万方程应为：

 （1）

其中郎之万力为白噪声，满足以下条件：



也即郎之万力均值为0，且前后不相关。

其中，为粘滞力系数， 为玻尔兹曼常数，为温度。



2）该系统动能可以表示为：



该系统势能可以表示为：



该系统的总能量可以表示为：



因此该能量平均则可以表示为：

 （2）

也即求解郎之万方程，再求取粒子位置和速度的自相关函数的时间平均即可（时间平均等于系综平均）。

对（1）式作傅里叶变换有：



可以分别解出

 （3）

 （4）

由“谱频率为自相关函数的傅里叶变换”出发。现在令：谱频率 为粒子位置的自相关函数的傅里叶变换；谱频率 为粒子速度的自相关函数的傅里叶变换；谱频率 为郎之万力的自相关函数的傅里叶变换。

由 有。

利用傅里叶逆变换，由（3）式有

（5）

利用傅里叶逆变换，由（4）式有

（6）

（这个地方没积出来，还在思考中~）

3）

由“时间平均等于系综平均”出发，对（1）式利用欧拉法和SRK2算法结合进行模拟：微分方程使用龙格库塔算法，随机微分方程部分使用随机龙格库塔算法。

在松弛一段时间后，取模拟的时间平均作为（2）式的估计值。欧拉法和SRK2算法与PPT给出的流程基本一致，这里不再赘述，其中，由涨落耗散定理可有 。

假定，模拟的时间为100s，松弛时间为20s。模拟的粒子能量平均值随温度 的变化规律如图1所示。

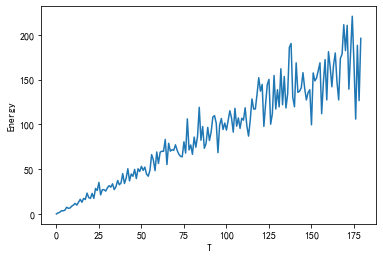


图1 粒子能量平均值随温度 的变化规律

代码（<https://github.com/Acpnohc/conputational_method_in_theory_physics/tree/main/hw2> ）在该网址下载运行。

**代码问题与思考**

方程（1）我认为实际上是一个 随机微分方程组。

1. 根据第二节课老师您说分两部分处理：第一部分用龙格库塔算法，第二部分用随机龙格库塔算法。但是实际上第一个方程似乎不适合用龙格库塔：因为二阶要向前递推一步，除非用插值，不然怎么递推？

2. 微分方程组的龙格库塔有另外的形式，不知是我对流程图的理解有误，我感觉得用方程组形式的，我还没有尝试，后续会和您继续讨论。

**参考文献**

———————————————————————————————————————

[1] Force calibration and noise in single-moleculemanipulation experiments

[2] Yaghoubi, Mohammad, et al. "Energetics of a driven Brownian harmonic oscillator." Journal of Statistical Mechanics: Theory and Experiment 2017.11 (2017): 113206.

———————————————————————————————————————

**代码附录**

———————————————————————————————————————

# -\*- coding: utf-8 -\*-

"""

Created on Mon Mar 22 19:03:14 2021

@author: JChonpca\_Huang

"""

import numpy as np

import matplotlib.pyplot as plt

def f\_sode(x,v):

return -x -v

def SRK2(fv,h,t,x0,v0,T):

XX = np.zeros(int(t/h)+1)

XX[0] = x0

VV = np.zeros(int(t/h)+1)

VV[0] = v0

noise = np.random.normal(loc=0,scale=1,size=int(t/h))\*np.sqrt(2\*T\*h)

for i in range(1,int(t/h)+1):

vk1 = fv(XX[i-1], VV[i-1])

vk2 = fv(XX[i-1], VV[i-1] + h\*vk1 + noise[i-1])

VV[i] = VV[i-1] + (1/2)\*h\*(vk1+vk2) + noise[i-1]

XX[i] = XX[i-1] + VV[i-1]\*h

return [XX,VV]

def E\_stat(P):

x = P[0][2000::]

v = P[1][2000::]

E = 0

for i in range(len(x)):

E = E + (1/2)\*(x[i]\*\*2 + v[i]\*\*2)

return E/len(x)

X = []

Y = []

P = []

for T in range(0,180):

p = SRK2(f\_sode,0.01,100,0,0,T)

P.append(p)

X.append(T)

Y.append(E\_stat(p))

plt.plot(X,Y)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

plt.xlabel("T")

plt.ylabel("Energy")

plt.show()