计算物理 A 第十题

杨旭鹏 PB17000234

2019 年秋季

目录

1 题目描述

Monte Carlo 方法研究正弦外力场 $F \propto sin(\omega t)$ 中的随机行走。

2 算法

2.1 随机行走公式推导

设粒子在 $t=0$ 时刻从原点出发，粒子每一步时间间隔为 Δt，正弦作用力为正值时代表作用在粒子上的力方向向右，负值表示向左。则第 n 步时粒子向右移动的概率为 $p_{right}(n\Delta t)=\frac{1}{2}(1+ksin(\omega n\Delta t))$，向左移动的概率为 $p_{left}(n\Delta t)=\frac{1}{2}(1-ksin(\omega n\Delta t))$。其中 $k\in[0,1]$ 为正弦力对粒子运动的影响因子，正弦力越大，对粒子运动的影响越大，相应地影响因子 k 越大。

若我们使用最简单的一维 Random Walk 模型，考虑每一步的步长为一定值，不妨假设每一步的步长为 $l=1$。则在第 n 步时，我们产生 $[0,1]$ 的随机数 ξ，若 $\xi\in[0,p_{left}(n\Delta t))$, 此时粒子向左移动，若 $\xi\in(p_{left}(n\Delta t),1]$, 则粒子向右移动。

2.2 16807 产生器

作用中所产生的随机数使用 16807 产生器产生。16807 产生器属于线性同余法产生器的特例。而线性同余法方法为:

$$\begin{aligned} I_{n+1} &= (aI_n+b)\ mod\ m \\ x_n &= I_n/m \end{aligned} \tag{1}$$

其中整数 $I_i\in[0,m-1]$，a,b,m 为算法中的可调参数，其选取直接影响产生器的质量。选取参数:

$$\begin{cases} a=7^5=16807 \\ b=0 \\ m=2^{31}-1=2147483647 \end{cases} \tag{2}$$

即为所谓的 16807 产生器。由于直接利用1编写程序时计算 $(aI_n\ mod\ m)$ 时很容易造成数据溢出，故采取 Schrage 方法进行具体编程的实现:

$$aI_n\ mod\ m=\begin{cases} a(I_n\ mod\ q)-r[I_n/q], & if\geq 0 \\ a(I_n\ mod\ q)-r[I_n/q]+m, & otherwise \end{cases} \tag{3}$$

其中 $m=aq+r$，即 $q=m/a=127773$，$r=m\ mod\ a=2836$。即可利用此方法产生伪随机数序列。

3 程序使用方法

在运行程序后，会看到请求输入所需计算的随机行走的总步数，按照提示在后面输入，摁回车继续。屏幕请求输入进行模拟的总粒子数，输入后摁回车继续。之后按照提示选择是否输出每个粒子的模拟结果，输入后摁回车继续。然后程序会输出总模拟粒子数的平均位置和距离原点平均距离的平方到数据文件。[1]程序输出完这些后会自动退出。

```
请输入您所需要的总步数：1000
请输入您所需总粒子个数：10
是否输出每个模拟粒子的位置数据至文件？是输1，不是输0:1
Program ended with exit code: 0
```

图 1: 一个典型程序的运行示例

4 程序结果与讨论

现在我们从理论上推导出粒子在正弦力场下的 $\langle x(N)\rangle$ 和 $\left\langle x^2(N)\right\rangle$。

$$\begin{aligned}\langle x(t)\rangle &= \left\langle \sum_{i=1}^{N} \Delta x(i) \right\rangle = \sum_{i=1}^{N} \langle \Delta x(i)\rangle \\ &= \sum_{i=1}^{N} k sin(\Delta t\omega i) = k\frac{sin(\frac{N\omega\Delta t}{2})sin(\frac{N+1}{2}\Delta t\omega)}{sin(\frac{\Delta t\omega}{2})} \\ &= k\frac{sin(\frac{t\omega}{2})sin((\frac{t+\Delta t}{2})\omega)}{sin(\frac{\Delta t\omega}{2})}\end{aligned} \tag{4}$$

[1]关于程序的参数可在程序宏定义中更改，包括影响因子 k，正弦力场的角频率 w，进行模拟的每一步的时间间隔 dt。默认值分别为 0.5,1,0.1

$$
\begin{aligned}
\left\langle x^2(t)\right\rangle &= \left\langle \left(\sum_{i=1}^{N}\Delta x(i)\right)^2 \right\rangle = \sum_{i=1}^{N}\left\langle \Delta x^2(i)\right\rangle + \sum_{i\neq j}^{N}\left\langle \Delta x(i)\Delta x(j)\right\rangle \\
&= N\Delta t + \sum_{i\neq j}^{N} k^2 sin(i\omega\Delta t)sin(j\omega\Delta t) \\
&= N + k^2\left(\sum_{i=1}^{N} sin(i\omega\Delta t)\sum_{j=1}^{N} sin(j\omega\Delta t) - \sum_{i=1}^{N} sin^2(i\omega\Delta t)\right) \\
&= N + k^2\left(\sum_{i=1}^{N} sin(i\omega\Delta t)\right)^2 - k^2\sum_{i=1}^{N} sin^2(i\omega\Delta t) \\
&= N + k^2\left(\frac{sin(\frac{N\omega\Delta t}{2})sin(\frac{N+1}{2}\omega\Delta t)}{sin(\frac{\omega\Delta t}{2})}\right)^2 - \frac{k^2}{2}\sum_{i=1}^{N}(1 - cos(2i\omega\Delta t)) \\
&= N\left(1 - \frac{k^2}{2}\right) \\
&+ k^2\left[\left(\frac{sin(\frac{N\omega\Delta t}{2})sin(\frac{N+1}{2}\omega\Delta t)}{sin(\frac{\omega\Delta t}{2})}\right)^2 + \frac{sin(N\omega\Delta t)cos(N+1)\omega\Delta t}{2sin(\omega\Delta t)}\right] \\
&= \frac{t}{\Delta t}\left(1 - \frac{k^2}{2}\right) \\
&+ k^2\left[\left(\frac{sin(\frac{\omega t}{2})sin(\frac{t+\Delta t}{2}\omega)}{sin(\frac{\omega\Delta t}{2})}\right)^2 + \frac{sin(\omega t)cos(t+\Delta t)\omega}{2sin(\omega\Delta t)}\right]
\end{aligned}
\tag{5}
$$

而 $var(x(N))$ 为：

$$
\begin{aligned}
var(x(t)) &= \left\langle x^2(t)\right\rangle - \left(\left\langle x(t)\right\rangle\right)^2 \\
&= \frac{t}{\Delta t}\left(1 - \frac{k^2}{2}\right) + k^2\frac{sin(t\omega)cos(t+\Delta t)\omega}{2sin(\Delta t\omega)}
\end{aligned}
\tag{6}
$$

以 $k = 0.5, \omega = 1, t = 0.1$ 为例，将上述理论分布画图得到：

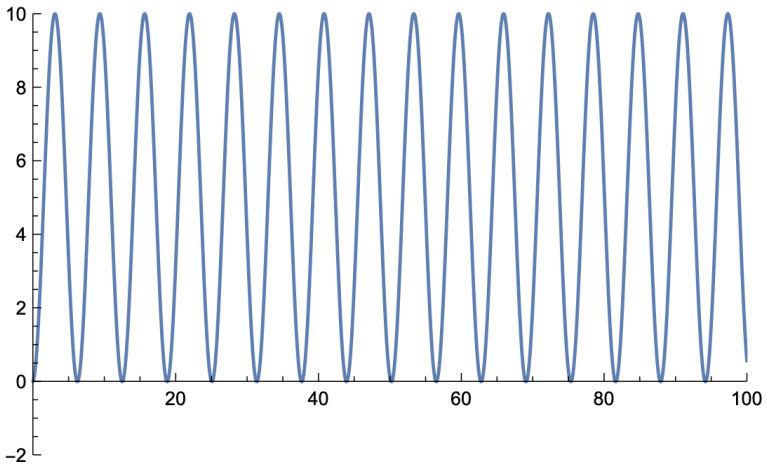


图 2: $k = 0.5, \omega = 1, t = 0.1$ 时 $< x(t) >$ 的理论曲线

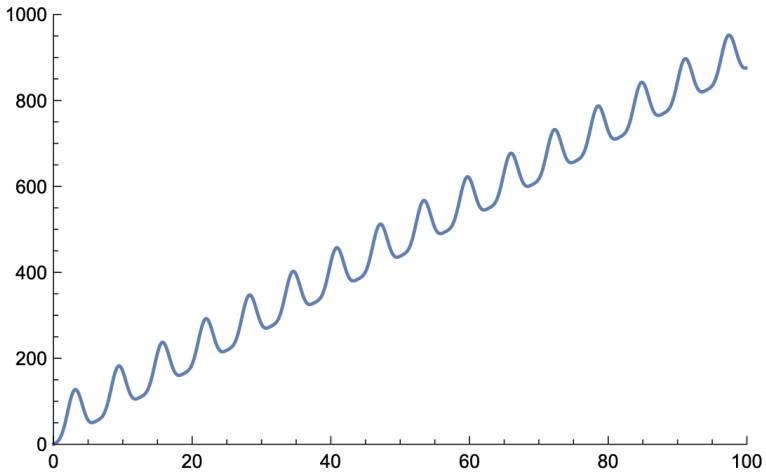
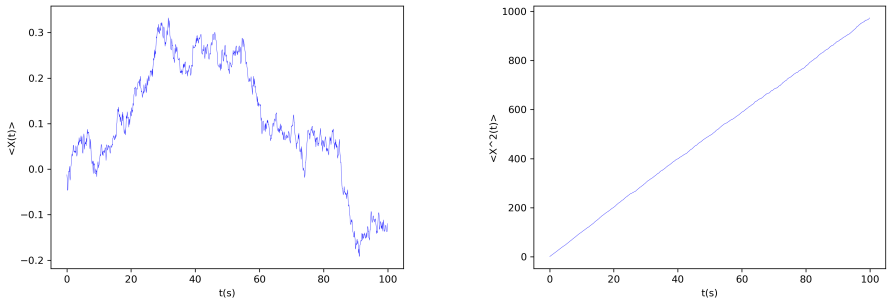


图 3: $k = 0.5, \omega = 1, t = 0.1$ 时 $< x^2(t) >$ 的理论曲线

当我们选取 $k = 0$ 时，此时相当于自由的一维布朗粒子，有程序结果：



(a) 粒子的平均位置　　(b) 粒子距离原点距离平方的平均

图 4: $k = 0$ 时模拟 10000 个粒子行走 1000 步

可以看出粒子的平均位置在模拟步数内，均比较接近于 0，表现出布朗粒子的特性。而从结果上也能看出距离原点距离平方的平均与步数成正比关系，验证了布朗粒子的特性。

对于模拟的 5 个粒子有：

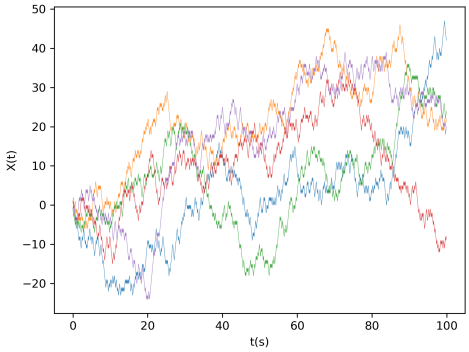
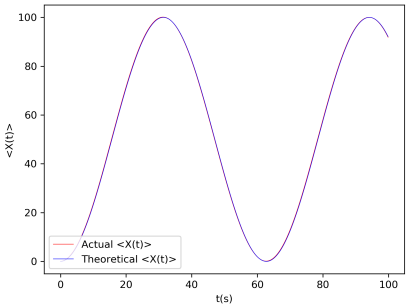
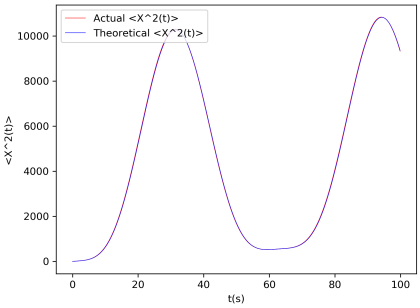


图 5: $k = 0$ 时 5 个粒子的运动位置模拟结果

当我们设定 $k = 0.5$ 时，有：

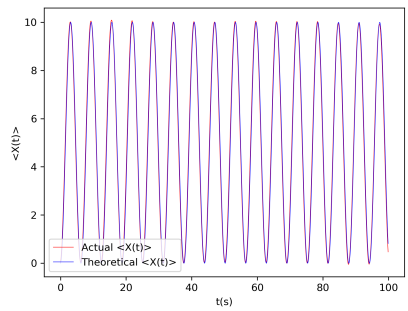


(a) 粒子的平均位置

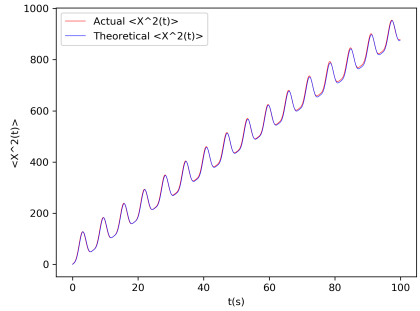


(b) 粒子距离原点距离平方的平均

图 6: $k = 0.5, \omega = 0.1$ 时模拟 100000 个粒子行走 1000 步

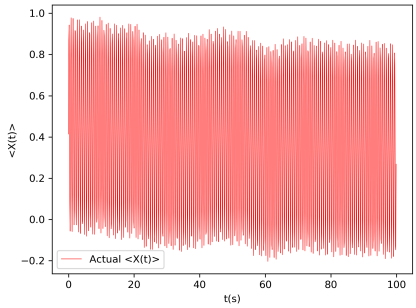


(a) 粒子的平均位置

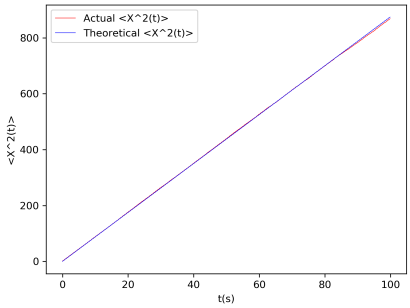


(b) 粒子距离原点距离平方的平均

图 7: $k = 0.5, \omega = 1$ 时模拟 100000 个粒子行走 1000 步

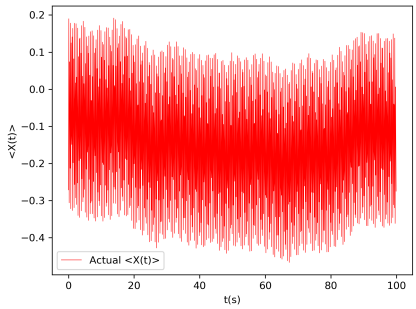


(a) 粒子的平均位置

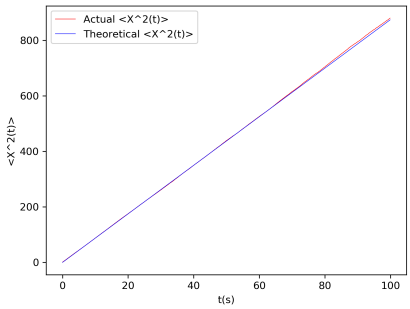


(b) 粒子距离原点距离平方的平均

图 8: $k = 0.5, \omega = 10$ 时模拟 100000 个粒子行走 1000 步



(a) 粒子的平均位置

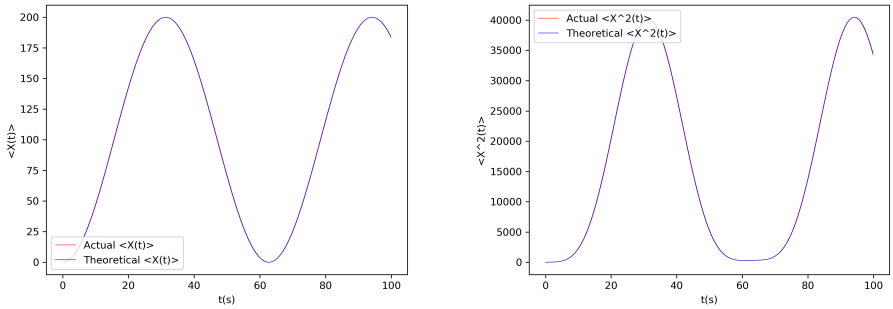


(b) 粒子距离原点距离平方的平均

图 9: $k = 0.5, \omega = 100$ 时模拟 100000 个粒子行走 1000 步

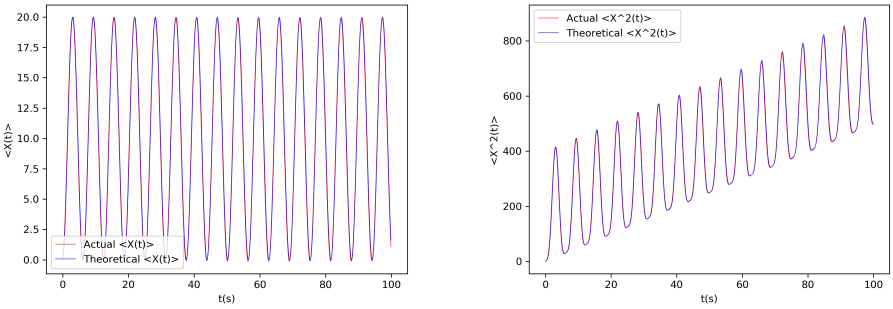
可以看出在影响因子 $k = 0.5$ 时粒子的平均位置已经呈现与正弦力同频率的震荡。而距离原点距离的平方的平均基本为在 $k = 0$ 时的线性关系上叠加与正弦力同频率的震荡。两者的震荡幅度均随着正弦力频率的增加而减少，这点也很好理解，当粒子受正弦力正周期的影响还不大时，就到了正弦力的负周期，从而低消正周期的影响。可以预见的是，当正弦力的周期增大时，粒子的运动行为越来越接近自由的布朗粒子。

当我们设置 $k = 1$ 时，得到：



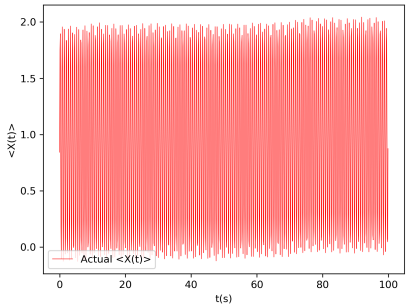
(a) 粒子的平均位置　　(b) 粒子距离原点距离平方的平均

图 10: $k = 1, \omega = 0.1$ 时模拟 100000 个粒子行走 1000 步

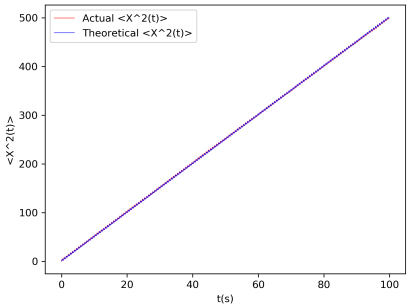


(a) 粒子的平均位置　　(b) 粒子距离原点距离平方的平均

图 11: $k = 1, \omega = 1$ 时模拟 100000 个粒子行走 1000 步

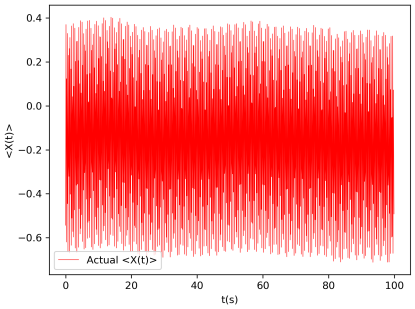


(a) 粒子的平均位置

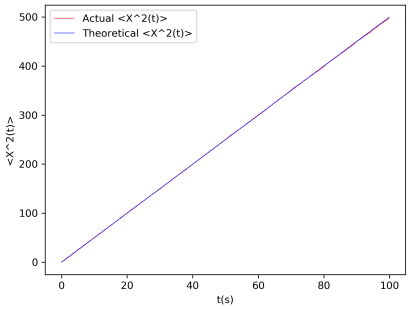


(b) 粒子距离原点距离平方的平均

图 12: $k = 1, \omega = 10$ 时模拟 100000 个粒子行走 1000 步



(a) 粒子的平均位置



(b) 粒子距离原点距离平方的平均

图 13: $k = 1, \omega = 100$ 时模拟 100000 个粒子行走 1000 步

可以看出在影响因子 $k=1$ 时粒子的平均位置已经呈现与正弦力同频率的震荡。而距离原点距离的平方的平均基本为在 $k=0$ 时的线性关系上叠加与正弦力同频率的震荡。两者的震荡幅度均随着正弦力频率的增加而减少，与 $k=0.5$ 时规律基本相同，但震荡幅度比同条件的 $k=0.5$ 时的要大，这是由于此时粒子的运动受正弦力场的影响大大增强造成。

若我们对 100000 个粒子行走到第 1000 步（取 $\Delta t=0.1$，则为 $t=100$ 时）的位置进行统计，得到：

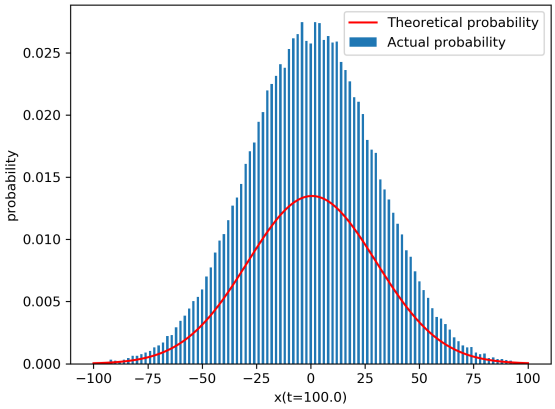


图 14: $k=0.5, \omega=1$ 时模拟 100000 个粒子行走 1000 步的位置分布统计

容易发现在奇数点的取值为 0，（因为假设每一步的步长为 1，故偶数步后粒子只能运动到偶数位置），故导致实际概率分布于理论概率分布差别较大。

为了修正此差别，我们将理论概率分布的值 $\times 2$，与实际概率分布对比得到：

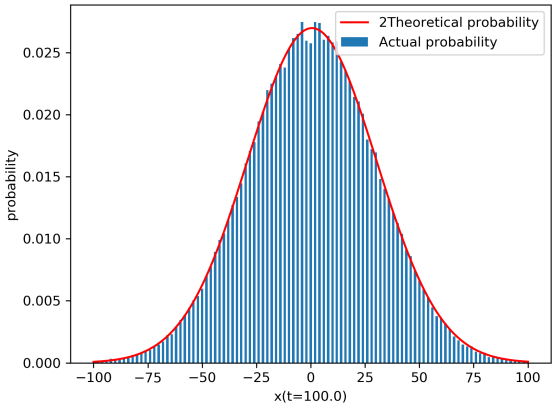


图 15: $k = 0.5, \omega = 1$ 时模拟 100000 个粒子行走 1000 步的位置分布统计

可发现两者差别比较小，基本验证位置分布。对于 t 时刻的粒子理论位置分布，有：

$$p(x(t)) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-<x(t)>)^2}{2\sigma^2}} \tag{7}$$

其中 σ^2 为 $x(t)$ 的方差。值得注意的是此理论分布是严格的二项分布的近似，不过这对于比较模拟结果足够了。严格概率分布在 11 题中也进行了讨论，这里不再展开叙述。

可以看出和计算模拟结果相差不大，差别可能是由于模拟离子不够多造成的统计涨落误差。

5 心得与体会

通过此次作业，对随机行走模型和布朗运动有了更深刻的认识。

6 附录

A C 语言源程序

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define a 16807
#define b 0
#define m 2147483647
#define r (m%a)
#define q (m/a)
#define Pi 3.1415926
#define dt 0.1 //每步间隔时间
#define k 0.5 //影响因子的大小
#define w 1 //正弦力场的角速度大小
#define len 100 //统计概率分布时的统计范围
#define round(x) ((x)>=0?(int)((x)+0.5):(int)((x)-0.5))


//利用/dev/random 产生"真"随机数
int my_realrandom(int ran[],int n){
    FILE *fp1 = fopen("/dev/random","r");
    for(int i=0;i<n;i++){
        fread(&ran[i], 1, 4, fp1);
    }
    fclose(fp1);
    return 0;
}



int my_filewriter_double(char str1[],char str2[],double num[],int
    n){
    FILE * fp;
    char str[20];
```

```
32      strcpy(str,str1);
33      strcat(str,str2);
34      fp = fopen(str,"w+");
35
36      for(int i=0;i<(n-1);i++)
37      {
38          fprintf(fp,"%lf,",num[i]);
39
40      }
41      fprintf(fp,"%lf",num[n-1]); //最后一个数据后不加 ","
42      fclose(fp);
43      return 0;
44  }
45
46
47
48  int num2str(char str[],int num){
49      int n, i = 0;
50      char tmp[20];
51      n = num % 10;
52      while (n>0 || num > 0)
53      {
54          tmp[i++] = n + '0';
55          num = (num - n) / 10;
56          n = num % 10;
57      }
58      tmp[i] = '\0';
59      for (i=0; i<=strlen(tmp)-1; i++)
60      {
61          str[i] = tmp[strlen(tmp)-i-1];
62      }
63      str[i] = '\0';
64      return 0;
65  }
66
67
68  // Schrage 方法RW
69  int my_schrage_RW(double ran[],double evenx[],double evenx2[],int
```

```
    seed,int n,int x){
70  if (seed >= 0) {
71      ran[0] = seed / (double) m;
72  } else {
73      ran[0] = (seed + m) / (double) m;
74  }
75  if(seed == m-1){
76      if(a >= b){   //由于Schrage方法只对z in
            (0,m-1)成立，故这里要讨论z == m-1的情况
77          seed = m + (b-a) % m;
78      }
79      else  seed = (b-a) % m;
80
81  }
82
83  else seed = ((a * (seed % q) - r * (seed / q)) + b % m ) % m;
        //递推式
84  if (ran[0] < 0.5*( 1 - k*sin(w*dt)) )
85      ran[0] = -1;
86  else ran[0] = 1;
87  evenx[0] += ran[0]/x;
88  evenx2[0] += pow(ran[0],2)/x;
89
90  for (int i = 1; i < n-1; i++) {
91      if (seed >= 0) {
92          ran[i] = seed / (double) m;
93      } else {
94          ran[i] = (seed + m) / (double) m;
95      }
96      if(seed == m-1){
97          if(a >= b){   //由于Schrage方法只对z in
                (0,m-1)成立，故这里要讨论z == m-1的情况
98              seed = m + (b-a) % m;
99          }
100         else  seed = (b-a) % m;
101
102     }
103     else seed = ((a * (seed % q) - r * (seed / q)) + b % m ) %
```

```
            m; //递推式
104         if (ran[i] < 0.5*( 1 - k*sin(w*(i+1)*dt)) )
105             ran[i] = ran[i-1]- 1;
106         else ran[i] = ran[i-1] + 1;
107         evenx[i] += ran[i]/x;
108         evenx2[i] += pow(ran[i],2)/x;
109     }
110
111
112     if (seed >= 0) {
113         ran[n-1] = seed / (double) m;
114     } else {
115         ran[n-1] = (seed + m) / (double) m;
116     }
117     if (ran[n-1] < 0.5*( 1 - k*sin(w*n*dt)) )
118         ran[n-1] = ran[n-2]- 1;
119     else ran[n-1] = ran[n-2] + 1;
120     evenx[n-1] += ran[n-1]/x;
121     evenx2[n-1] += pow(ran[n-1],2)/x;
122
123     return 0;
124 }
125
126
127
128
129
130
131 int main(int argc, const char * argv[]) {
132     int N;   //总随机数个数(总步长k个数)
133     int n;   //总粒子个数
134     int flag = 0; //是否输出每个模拟粒子每步的位置数据文件
135     char str[50];
136     printf("请输入您所需要的总步数: ");
137     while (!scanf("%d",&N)){ //简单的输入检查
138         gets(str);
139         printf("\nInput error,please try again\n");
140         printf("请输入您所需要的总步数: ");
```

```
141     }
142
143     printf("请输入您所需总粒子个数: ");
144     while (!scanf("%d",&n)){ //简单的输入检查
145         gets(str);
146         printf("\nInput error,please try again\n");
147         printf("请输入您所需要的总粒子个数: ");
148     }
149
150     printf("是否输出每个模拟粒子的位置数据至文件？是输1，不是输0:");
151     while (!scanf("%d",&flag)){ //简单的输入检查
152         gets(str);
153         printf("\nInput error,please try again\n");
154         printf("是否输出每个模拟粒子的位置数据至文件？是输1，不是输0:");
155     }
156
157     if(N*n >1000000)
            printf("您输入的参数已接受，正在计算请稍等片刻~\n");
158
159     double *ran = malloc(sizeof(double) * N);
            //用来存放每个粒子每一步的位置
160     double *evenx = malloc(sizeof(double) * N);
            //用来存放所有粒子每一步位置的平均值
161     double *evenx2 = malloc(sizeof(double) * N);
            //用来存放所有粒子每一步位置平方的平均值
162     double *p = malloc(sizeof(double)*(2*len+1));
163     double *var = malloc(sizeof(double)*N);
164     int *seed = malloc(sizeof(int) * n); //用来存放随机种子值
165
166     my_realrandom(seed, n);
167
168
169     for(int i=0;i<N;i++){ //对evenx进行初始化
170         evenx[i] = 0;
171         evenx2[i] = 0;
172         var[i] = 0;
173     }
174
```

```
175     for(int i=0;i <= 2*len;i++){ //对evenx进行初始化
176         p[i] = 0;
177     }
178 
179 
180     char str1[10];
181     for(int i = 0;i<n;i++){
182         num2str(str1,i+1);
183         my_schrage_RW(ran,evenx,evenx2,seed[i],N,n);
184         p[round(ran[N-1]) + len] += (double)1/n;
185         if(flag == 1 && n < 20 ) my_filewriter_double(str1,
                "RW.dat", ran, N);
186     }
187     num2str(str1, n);
188     for(int i=0;i<N;i++){
189         var[i] = (evenx2[i]-pow(evenx[i],2))/n;
190     }
191     my_filewriter_double(str1, "evenx.dat", evenx, N);
192     my_filewriter_double(str1, "evenx2.dat", evenx2, N);
193     my_filewriter_double(str1, "p.txt", p, (2*len+1));
194     my_filewriter_double(str1, "var.txt", var, N);
195     return 0;
196 }
```

B 可视化绘图 python 程序源码

```
1
2  import matplotlib.pyplot as plt
3  from mpl_toolkits.mplot3d import Axes3D
4  import numpy as np
5  #from IPython.core.pylabtools import figsize # import figsize
6  #figsize(12.5, 4) # 设置 figsize
7  plt.rcParams['savefig.dpi'] = 300 #图片像素
8  plt.rcParams['figure.dpi'] = 300 #分辨率
9  # 默认的像素: [6.0,4.0], 分辨率为100, 图片尺寸为 600&400
10 fig1 = plt.figure()
11 fig2 = plt.figure()
12 fig = plt.figure()
13 pfig = plt.figure()
14 figvar = plt.figure()
15 ax1 = fig1.add_subplot(111)
16 ax2 = fig2.add_subplot(111)
17 ax = fig.add_subplot(111)
18 pax = pfig.add_subplot(111)
19 axvar = figvar.add_subplot(111)
20
21 n = 5
22 N = 1000
23 dt = 0.1
24 w = 1
25 k = 0
26
27 X = []
28 Y1 = []
29 Y2 = []
30 p = []
31 var = []
32
33 with open('problem 10/w='+str(w)+'/'+str(N)+'evenx_'+str(k)+'.dat',
       'r') as f:
34     while True:
```

```
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        Y1 = [float(i) for i in lines.split(',')] # 将整行数据分割处理
    Y1 = np.array(Y1) # 将数据从list类型转换为array类型。

with open('problem
    10/w='+str(w)+'/'+str(N)+'evenx2_'+str(k)+'.dat', 'r') as f:
    while True:
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        Y2 = [float(i) for i in lines.split(',')] # 将整行数据分割处理
    Y2 = np.array(Y2) # 将数据从list类型转换为array类型。


with open('problem 10/'+str(100000)+'p.txt', 'r') as f:
    while True:
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        p = [float(i) for i in lines.split(',')] # 将整行数据分割处理
    p = np.array(p) # 将数据从list类型转换为array类型。


with open('problem 10/'+str(10000)+'var.txt', 'r') as f:
    while True:
        lines = f.readline() # 整行读取数据
        if not lines:
            break
        var = [float(i) for i in lines.split(',')] #
            将整行数据分割处理
    var = np.array(var) # 将数据从list类型转换为array类型。


Y2 = np.sqrt(Y2*Y2)
X = np.arange(0, N * dt, dt)
px = np.arange(-100, 101, 1)
```

```
varx = np.arange(0, 10000, 1)
print(np.shape(px))
py = 1/np.sqrt(2*np.pi*875.265)*np.exp(-np.power(px-0.56124,
    2)/(2*875.265))


ax1.plot(X, Y1, 'b', label='<X(t)>', lw=0.3)
ax1.set_xlabel('t(s)')
ax1.set_ylabel('<X(t)>')
fig1.savefig("1.png")

axvar.plot(varx, var, 'b', label='<X(t)>', lw=0.3)
axvar.set_xlabel('t(s)')
axvar.set_ylabel('var(t)')
figvar.savefig("var.png")


ax2.plot(X, Y2, 'b', label='<X^2(t)>', lw=0.3)
ax2.set_xlabel('t(s)')
ax2.set_ylabel('<X^2(t)>')
fig2.savefig("2.png")

pax.bar(px, p, width=1.1, label='Actual probability')
pax.plot(px, py, 'r', label='Theoretical probability')
pax.set_xlabel('x(t='+str(N*dt)+')')
pax.set_ylabel('probability')
pax.legend(loc=1)
pfig.savefig("p.png")

Y = [[]]
Y = np.array(Y)

for i in range(n):
    with open('problem 10/'+str(i+1)+'RW.dat', 'r') as f:
        while True:
            lines = f.readline() # 整行读取数据
            if not lines:
                break
```

```
108             Y = np.append(Y, [[float(i) for i in lines.split(',')]])
                    # 将整行数据分割处理
109
110 Y = Y.reshape(n, N)
111
112 plt_label = 0
113 for link in range(n):
114     ax.plot(X, Y[link], label='particle'+str(plt_label), lw=0.3)
115     plt_label += 1
116
117 ax.set_xlabel('t(s)')
118 ax.set_ylabel('X(t)')
119 fig.savefig("3.png")
```