计算物理作业十一

于浩然 PB19020634 2021.11.12

1 作业题目

计算 2 维正方格子中 GSAW 的指数值，并定性地加以讨论. 进一步，如何研究球面网格上的 SAW，它与平面上的结果会有什么不同?

2 算法简介

2.1 生长性自规避随机行走

在粒子的自规避行走 SAW(*self-avoiding walks*) 模型中，粒子要记住以往走过格点的位置，在禁止返回上一位置的同时，若轨迹与历史轨迹相交时粒子将死亡从而停止行走. SAW 的禁止性记忆迫使路径范围扩大，与 RW 模型有着显著区别.

由于 SAW 模型中随机行走粒子易与自身轨迹交叠而死去，因此损耗较大，于是人们设计了一种避免死亡的自规避行走，称为生长性自规避行走 (GSAW)，在这个模型中，粒子会记住以前走过的位置，尽可能避免走进死胡同而选择其他安全的路径.

2.2 指数的计算

一般 RW 前后距离的方均值为

$$\langle r^2(N)\rangle = aN^{2\nu}(1 + bN^{-\Delta} + \cdots) \tag{1}$$

其中指数 ν 说明了方均根值如何随 N 趋于无穷大，ν 越大则方均根增长越快，可视为相变标度率下的临界指数，Δ 是修正标度的指数 (小量). 此外还可定义与相变问题中配分函数等价的物理量

$$Z_N \propto N^{\gamma-1}q_{eff}^N, \tag{2}$$

它描述格点上不同随机行走数目随行走步数的变化关系，q_{eff} 为有效坐标数.

由于自规避的排斥效应，尽管 SAW 结果的路径范围比理想的 RW 要大，但是 N 大对应的路径数目较少. 由 (1) 式，我们通过双对数曲线 $\log\langle r^2(N)\rangle - \log N$ 求斜率来求指数 ν:

$$\nu(N) = \frac{1}{2}\frac{\ln[\langle r^2(N+i)\rangle/\langle r^2(N-i)\rangle]}{\ln[(N+i)/(N-i)]} \tag{3}$$

式中 $i \ll N$，使它足够大能够忽略 N 附近的涨落，但又要比 N 小很多以使修正项 $N^{-\Delta}$ 对指数的计算影响很小. 类似地，对 (2) 式两侧取对数:

$$\ln Z(N) = (\gamma - 1)\ln N + N\ln q_{eff} \tag{4}$$

为了消去 q_{eff} 项，分别取 $N-i, N+i, N$ 项进行线性组合，易得下式：

$$\begin{aligned} & \ln[Z(N)/Z(N-i)] - \ln[Z(N+i)/Z(N)] \\ = \ & (\gamma-1)\ln[N^2/(N-i)(N+i)] \approx (\gamma-1)(i/N)^2 \end{aligned} \tag{5}$$

$$\gamma(N) = 1 + \left(\frac{N}{i}\right)^2 \ln \frac{Z^2(N)}{Z(N-i)Z(N+i)} \tag{6}$$

同样，i 应足够大而能够忽略 N 附近的统计涨落.

3 编程实现

使用 FORTRAN90 进行编程

3.1 求指数 ν 值

用函数 `sqdr(steps)` 生成指定步数随机行走并输出其 RW 距离的平方 `sqdr`，再在子程序 `Getnu()` 中循环引用上面的函数来求不同 N 值对应 ν 指数

3.2 求指数 γ 值

函数 `Z(total, steps)` 求 `total` 粒子数中达到步数 `steps` 的粒子所占比重，可以认为此值与公式 (6) 中 $Z(N)$ 成正比. 随后在子程序 `Getgamma()` 中实现求不同 N 值对应的 γ 指数.

3.3 参数选取

选取 N 从 50 到 200，$i = 10$，运行程序. 随后用 python 脚本实现画图. 程序中，粒子能够很大程度上避免一步之内的错误：粒子将自动跳过上一步来时的方向 (见 `FUNCTION noturn`)；当抽样结果将导致粒子直接在下一步死亡 (自相交) 时，将会重新进行抽样，当抽样若干次仍不能逃生时死亡. 这样便实现了一定的 Growing 功能.

4 计算结果

4.1 求指数 ν 值

将计算所得的 ν 指数展示如下：

计算结果有一定统计涨落，其中心值在 0.7 左右，这与一般 (非自规避)RW 的值 ($\nu = 0.5$) 相比要更大，也就是说随着 N 增长，GSAW 比普通 RW 相比 $\langle r^2(N)\rangle$ 的值变大的更快，这与理论和直觉相符合.

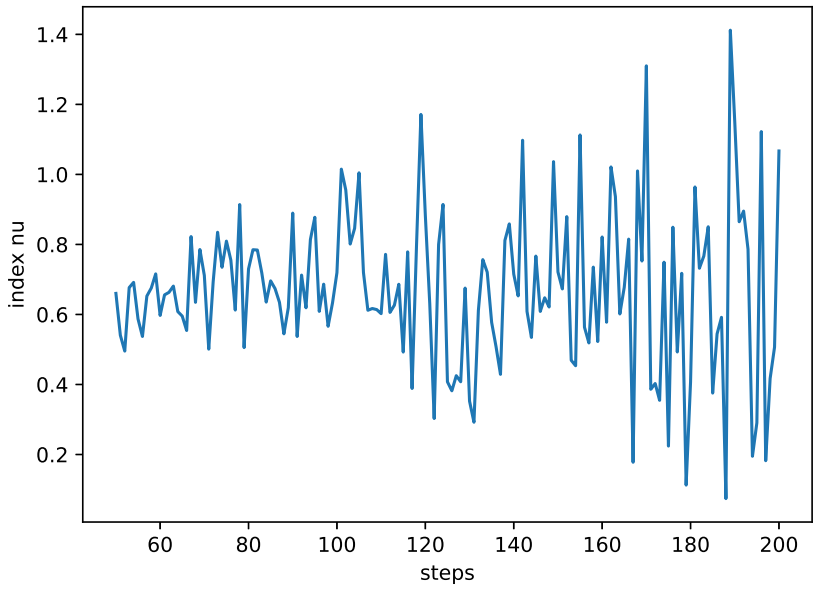

图 1: $N-\nu$ 关系图

4.2 求指数 γ 值

将计算出的 γ 指数展示如下:

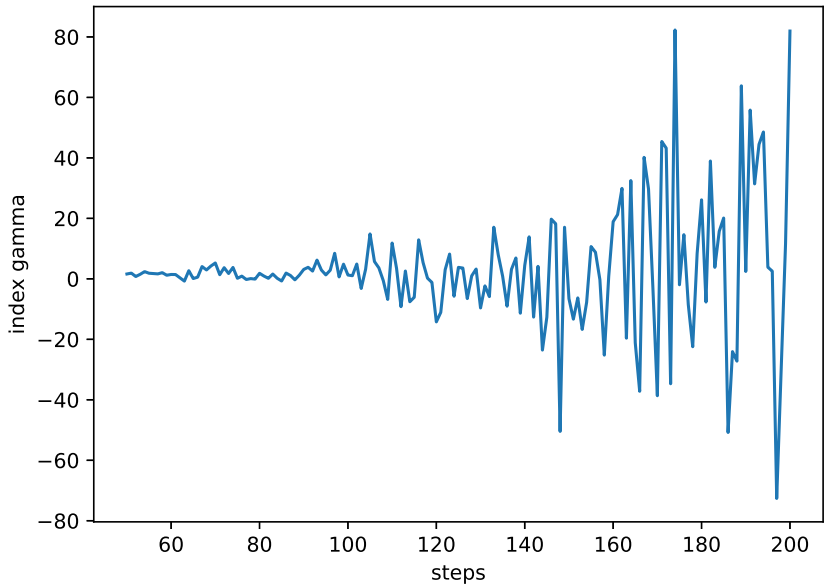

图 2: $N-\gamma$ 关系图

结果显示，指数 γ 值有较大的起伏，在 0 附近波动且随着 N 增大波动幅度越来越大. 这在一定程度上表明行走步数越大，一定步数所对应的行走数目与步数的关系

越不确定，增加或减少步数对随机行走数的影响越大，这可以理解为是 SAW 的约束性所导致的.

5 结论

本题中我们一定程度上实现了 GSAW 的功能，并计算了相关指数与一般 RW 比较，得出了 GSAW 的一些相关性质.

6 源代码

FORTRAN90 程序显示如下：

```
MODULE RW
IMPLICIT NONE

CONTAINS
FUNCTION sqdr(steps) ! RW生成器函数，返回行走距离的平方sqdr
    INTEGER(KIND=4), INTENT(IN) :: steps
    INTEGER(KIND=4) :: i, j
    INTEGER(KIND=4) :: dir, lastdir, tmpr(2)
    INTEGER(KIND=4), DIMENSION(0:steps, 2) :: r
    INTEGER(KIND=4), DIMENSION(-steps:steps, -steps:steps) ::
       lattice
    REAL(KIND=8) :: sqdr
    LOGICAL(KIND=4) :: revive, reach
    REAL(KIND=8) :: seed, rand(steps), tmprand(50)

    reach = .FALSE.
    DO WHILE(reach .EQV. .FALSE.)
        CALL RANDOM_NUMBER(seed) !用FORTRAN自带的随机数生成器生
           成16807生成器的种子
        CALL Schrage(steps, int(2147483647 * seed), rand)
        r(0, :) = [0, 0]
        lattice = 0
        lattice(0, 0) = 1
        DO i = 1, steps
            IF(i .EQ. 1) THEN ! 第一步有4个可能方向
                dir = int(4 * rand(i)) ! 设dir=0,1,2,3分别对应向
                   左,下,右,上
```

```
25                r(i, :) = r(i-1, :) + walk(dir)
26                lattice(r(i, 1), r(i, 2)) = 1 ! 将走过点对应的
                     lattice值标记为1
27                lastdir = dir
28            ELSE ! 后面的步最多只能有3个可能方向
29                dir = noturn(lastdir, int(3 * rand(i)))
30                ! 从除了上一步行走方向反方向的3个选择中随机选择
                     一个
31                tmpr(:) = r(i-1, :) + walk(dir) ! tmpr表示粒子下
                     一步可能前往的位置
32                IF(lattice(tmpr(1), tmpr(2)) .EQ. 1) THEN
33                    revive = .FALSE.
34                    CALL RANDOM_NUMBER(seed)
35                    CALL Schrage(20, int(2147483647 * seed),
                         tmprand)
36                    ! 若抽中的方向将导致粒子死亡, 再次抽样尽量避
                         免这次死亡
37                    DO j = 1, 50
38                        dir = noturn(lastdir, int(3 * tmprand(j)
                             ))
39                        tmpr(:) = r(i - 1, :) + walk(dir)
40                        IF(lattice(tmpr(1), tmpr(2)) .NE. 1)
                             THEN
41                            r(i, :) = tmpr(:)
42                            revive = .TRUE.
43                            EXIT
44                        END IF
45                    END DO
46                    IF(revive .EQV. .FALSE.) THEN
47                        EXIT ! 如果粒子复活失败, 则死亡
48                    END IF
49                ELSE
50                    r(i, :) = tmpr(:)
51                    lattice(r(i, 1), r(i, 2)) = 1
52                END IF
53                lastdir = dir
54            END IF
55            IF(i .EQ. steps) THEN
56                reach = .TRUE.
```

```
57              END IF
58          END DO
59      END DO
60      sqdr = DOT_PRODUCT(r(steps, :), r(steps, :)) ! 求走过了指定
            步数RW距离的平方
61  END FUNCTION sqdr
62
63  FUNCTION Z(total, steps) ! 同为RW生成器函数, 返回达到某指定步数
        粒子的比率
64      INTEGER(KIND=4), INTENT(IN) :: total, steps
65      INTEGER(KIND=4) :: successes, l, i, j
66      INTEGER(KIND=4) :: dir, lastdir, tmpr(2)
67      INTEGER(KIND=4), DIMENSION(0:steps, 2) :: r
68      INTEGER(KIND=4), DIMENSION(-steps:steps, -steps:steps) ::
            lattice
69      LOGICAL(KIND=4) :: revive
70      REAL(KIND=8) :: seed, rand(steps), tmprand(50), Z
71
72      successes = 0
73      DO l = 1, total
74          CALL RANDOM_NUMBER(seed) !用FORTRAN自带的随机数生成器生
                成16807生成器的种子
75          CALL Schrage(steps, int(2147483647 * seed), rand)
76          r(0, :) = [0, 0]
77          lattice = 0
78          lattice(0, 0) = 1
79          DO i = 1, steps
80              IF(i .EQ. 1) THEN ! 第一步有4个可能方向
81                  dir = int(4 * rand(i)) ! 设dir=0,1,2,3分别对应向
                        左,下,右,上
82                  r(i, :) = r(i-1, :) + walk(dir)
83                  lattice(r(i, 1), r(i, 2)) = 1 ! 将走过点对应的
                        lattice值标记为1
84                  lastdir = dir
85              ELSE ! 后面的步最多只能有3个可能方向
86                  dir = noturn(lastdir, int(3 * rand(i)))
87                  ! 从除了上一步行走方向反方向的3个选择中随机选择
                        一个
88                  tmpr(:) = r(i-1, :) + walk(dir) ! tmpr表示粒子下
```

```
                    一步可能前往的位置
89                  IF(lattice(tmpr(1), tmpr(2)) .EQ. 1) THEN
90                      revive = .FALSE.
91                      CALL RANDOM_NUMBER(seed)
92                      CALL Schrage(20, int(2147483647 * seed),
                            tmprand)
93                      ! 若抽中的方向将导致粒子死亡，再次抽样尽量避
                            免这次死亡
94                      DO j = 1, 50
95                          dir = noturn(lastdir, int(3 * tmprand(j)
                                ))
96                          tmpr(:) = r(i - 1, :) + walk(dir)
97                          IF(lattice(tmpr(1), tmpr(2)) .NE. 1)
                                THEN
98                              r(i, :) = tmpr(:)
99                              revive = .TRUE.
100                             EXIT
101                         END IF
102                     END DO
103                     IF(revive .EQV. .FALSE.) THEN
104                         EXIT ! 如果粒子复活失败，则死亡，进入下
                                次循环
105                     END IF
106                 ELSE
107                     r(i, :) = tmpr(:)
108                     lattice(r(i, 1), r(i, 2)) = 1
109                 END IF
110                 lastdir = dir
111             END IF
112             IF(i .EQ. steps) THEN
113                 successes = successes + 1 ! 成功走到最后则累计成
                        功数successes
114             END IF
115         END DO
116     END DO
117     Z = real(successes) / total ! 计算成功比率
118 END FUNCTION Z
119
120 SUBROUTINE Getnu() ! 计算nu的子程序
```

```
121     INTEGER(KIND=4) :: j, N, minim = 50, maxim = 200
122     REAL(KIND=8) ,DIMENSION(50:200) :: nu
123     REAL(KIND=8) :: big_ave = 0, small_ave = 0
124     DO N = minim, maxim
125         print *, 'computing nu, ',N, 'steps / 200steps'
126         DO j = 1, 300
127             small_ave = small_ave - small_ave / j + sqdr(N-10) /
                    j
128             big_ave = big_ave - big_ave / j + sqdr(N+10) / j
129             ! 分别累计r(N-i),r(N+i)的平均值, 且防止溢出
130         END DO
131         nu(N) = 0.5 * LOG(big_ave / small_ave) / LOG(real(N+10)
               / real(N-10))
132         ! 按照公式由双对数曲线斜率求指数nu(N)
133     END DO
134     OPEN (1, file='nu.dat') !将nu(N)数值写入文件
135     WRITE (1, *) nu
136     CLOSE (1)
137     print *, 'Done with nu!'
138     print *, '-----------------------------------'
139 END SUBROUTINE Getnu
140
141 SUBROUTINE Getgam() ! 计算指数gamma的子程序
142     REAL(KIND=8), DIMENSION(50:200) :: gam
143     INTEGER(KIND=4) :: N, minim = 50, maxim = 200
144
145     DO N = minim, maxim
146         print *, 'computing gamma, ', N, 'steps / 200steps'
147         gam(N) = 1 + (real(N) / 10)**2 * LOG(Z(10000, N)**2 / &
148                 (Z(10000, N-10) * Z(10000, N+10)))
149     END DO
150     OPEN (1, file='gamma.dat')
151     WRITE (1, *) gam
152     CLOSE (1)
153     print *, 'done with gamma!'
154 END SUBROUTINE Getgam
155
156 FUNCTION noturn(lastd, r) !不走回上一步的方向选择器
157     INTEGER(KIND=4) :: r, lastd
```

```
158     INTEGER(KIND=4) :: noturn
159     SELECTCASE(r)
160         CASE(0)
161             noturn = MOD(MOD(lastd + 2, 4) + 1, 4)
162         CASE(1)
163             noturn = MOD(MOD(lastd + 2, 4) + 2, 4)
164         CASE(2)
165             noturn = MOD(MOD(lastd + 2, 4) + 3, 4)
166     END SELECT
167     ! 一个两个相反方向对应数字的关系为: A=MOD(B+2, 4)
168 END FUNCTION noturn
169 FUNCTION walk(direction) ! 将数字代号转换成位移坐标的函数
170     INTEGER(KIND=4) :: direction
171     INTEGER(KIND=4), DIMENSION(2) :: walk
172     SELECTCASE(direction)
173         CASE(0)
174           walk = [-1, 0]
175         CASE(1)
176           walk = [0, -1]
177         CASE(2)
178           walk = [1, 0]
179         CASE(3)
180           walk = [0, 1]
181     END SELECT
182 END FUNCTION walk
183
184 END MODULE RW
185
186 SUBROUTINE Schrage(num, z0, rand)
187     !Schrage随机数生成器子程序,将均匀随机数序列存放在数组rand中
188     IMPLICIT NONE
189     INTEGER(KIND=4) :: N = 1, num
190     INTEGER :: m = 2147483647, a = 16807, q = 127773, r = 2836,
            In(num), z0
191     REAL(KIND=8), INTENT(INOUT) :: rand(num)
192     In(1) = z0 !将传入值z0作为种子
193     rand(1) = REAL(In(1))/m
194     DO N = 1, num - 1
195         In(N + 1) = a * MOD(In(N), q) - r * INT(In(N) / q)
```

```
196         IF (In(N + 1) < 0) THEN !若值小于零, 按Schrage方法加m
197             In(N + 1) = In(N + 1) + m
198         END IF
199         rand(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
200     END DO
201
202 END SUBROUTINE Schrage
203
204 PROGRAM MAIN
205     USE RW
206     IMPLICIT NONE
207     CALL Getnu() ! 调用求指数nu的子程序
208     CALL Getgam() ! 调用求指数gamma的子程序
209 END PROGRAM MAIN
```