计算物理作业十三

于浩然 PB19020634 2021.12.06

1 作业题目

用 Metropolis-Hasting 抽样方法计算积分：

$$I = \int_0^\infty (x - \alpha\beta)^2 f(x) \mathrm{d}x = \alpha\beta^2 \tag{1}$$

$$f(x) = \frac{1}{\beta\Gamma(\alpha)} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp(-x/\beta) \tag{2}$$

设积分的权重函数分别为：$p(x) = f(x)$ 和 $p(x) = (x - \alpha\beta)^2 f(x)$.

给定参数 α, β，并用不同的 γ 值，分别计算积分，讨论计算精度和效率.

2 算法简介

2.1 Markov 链

当满足条件 $p_n(x_1, \cdots, x_n; t_1, \cdots, t_n) = p_n(x_1, \cdots, x_n; t_1 + \Delta t, \cdots, t_n + \Delta t)$ 时，称随机过程是平稳的，即时间的起点选取没有影响，时，称随机过程是平稳的，即时间的起点选取没有影响，时间序列仅表示 x 取值的先后.

我们称一个平稳的随机序列是 Markov 的，如果某一时刻 x 取值的条件几率独立于上一时刻之前所有 x 值的话. Markov 链的极限分布与初始分布的选择无关，仅决定于转移概率. 分布概率和转移概率分别用 p, W 来表示，达到平衡时有如下关系：

$$p_i = \sum_j p_j W_{ji} \tag{3}$$

2.2 平稳态的主方程

平稳态主方程可以写为

$$\frac{p(x)}{p(x')} = \frac{W(x' \to x)}{W(x \to x')} \tag{4}$$

上式中已经考虑了到达平衡态与时间无关. 此即统计力学中的细致平衡解. 将变量 x, x' 分别用下标 i, j 替换，则方程 (4) 写为

$$p_i W_{ij} = p_j W_{ji} \tag{5}$$

2.3 Metropolis-Hasting 抽样规则

Metropolis-Hasting 抽样规则中，建议分布和接受几率均取非对称，接受几率根据待满足的几率分布 p 而定.

由细致平衡条件，

$$\frac{p_j}{p_i}=\frac{W_{ij}}{W_{ji}}=\frac{T_{ij}A_{ij}}{T_{ji}A_{ji}},\quad A_{ij}=\min\left\{1,\frac{p_jT_{ji}}{p_iT_{ij}}\right\} \tag{6}$$

$$W_{ij}=\begin{cases}T_{ij} & ,p_jT_{ji}>p_iT_{ij}\\ T_{ij}(p_j/p_i) & ,p_jT_{ji}<p_iT_{ij}\end{cases} \tag{7}$$

式中 i,j 不相同，W_{ii} 须通过归一性求出. 根据建议分布 T 进行初始抽样，$T(x\to x')$ 最好是和 p 形状相近的任意条件概率分布. 不妨取 $T_{ij}=T(x\to x')=T(x')=0.5\exp\{-x'/\gamma\}$ 并设 $x_0=1$，具体抽样方法如下：

假设已经产生了 n 个抽样点 $x_1,x_2,\cdots,x_n$，由所设的建议分布 T_{ij} 给出试探解 x'.

(1) 做抽样 $x'=-\gamma\ln R$, R 是 (0,1) 区间上均匀分布的随机数；

(2) 计算 $r=\frac{p_jT_{ji}}{p_iT_{ij}}$,

具体到积分 1:

$$r=\left(\frac{x'}{x_n}\right)^{\alpha-1}\exp\{-(x'-x_n)/\beta\}\exp\{(x'-x_n)/\gamma\} \tag{8}$$

积分 2：没有意义. 这将在后面进行阐释.

$$x_{n+1}=\begin{cases}x' & ,if\ R<\min(1,r)\\ x_n & ,if\ R>\min(1,r)\end{cases} \tag{9}$$

(3) 积分 1:

$$I=\frac{1}{N}\sum_{i=1}^{N}(x_i-\alpha\beta)^2 \tag{10}$$

积分 2:

前面的方法总会得到相同的结果 1，Metropolis-Hasting 方法不能计算积分. 若要考虑对求积分方法作修改，序列 $\{x\}$ 反映了 x 按 $p(x)$ 分布的情况，积分 $\int_0^\infty p(x)\mathrm{d}x$ 可由 $p(x)$ 与 x 轴所围面积来反映，这相当于没有权重计算积分的情形，在考虑 Metropolis 方法的本题中便不予考虑.

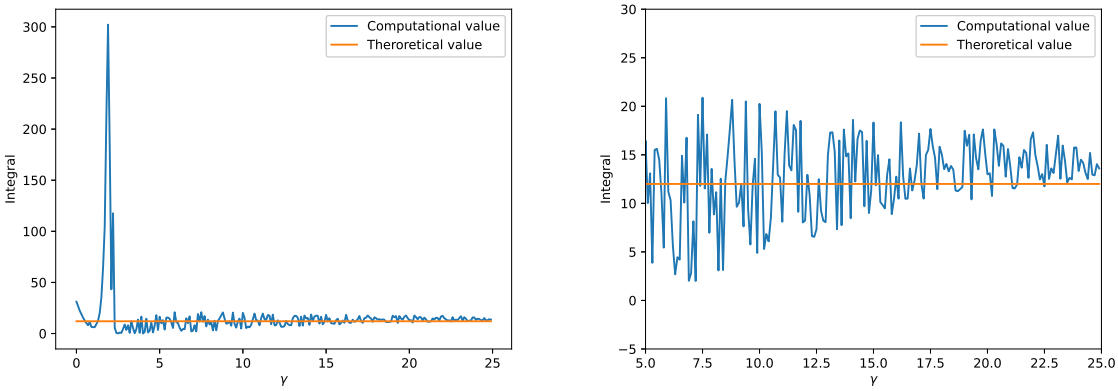
3 编程实现

不妨取参数 $\alpha=3,\beta=2$，积分理论值为 12. 求积分 1 时，γ 从 0 到 25 之间取，步进为 0.1；求积分 2 的子程序已编写，与积分一大同小异，但由于其没有意义故没有放在主程序中输出结果. 用 FORTRAN90 进行编程实现，具体程序见附录.

4 计算结果

- $p(x) = f(x)$

  积分值随 γ 值变化的情况如下：



(a) 全部值　　(b) 去除大偏差部分

图 1: 积分计算值随 γ 变化情况 (1)

图中看出，在 $\gamma = 2$ 附近有一极大的偏离，将 $\gamma < 5$ 的数据剔除后得到图 1(b). 作误差分布图如下：

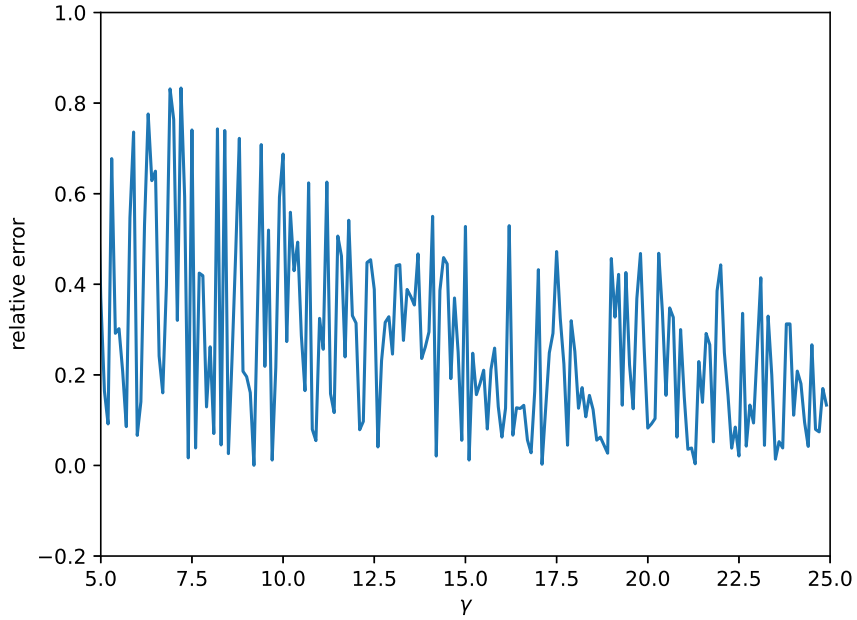


图 2: 积分 1 误差分布图

观察图像得到，在研究区间内误差随着 γ 增大而减小，当 $\gamma > 20$ 时误差情况较好，约在 30% 以内.

- $p(x) = (x - \alpha\beta)^2 f(x)$

  此时积分的计算无意义，在上一节中解释过，故不列出结果.

5 结论

本次作业实现了 Metropolis-Hasting 抽样方法计算积分，得到了具有一定误差的结果，当 γ 为适当值时积分计算更加精确，推测这是由于初始分布的形状与实际分布比较接近. 并发现了当除去权重函数的被积函数为常数时这一方法将不再有效，总得到无意义的结果 “1” . 推测这是由于在使用权重函数时仅顾及了分布而丢失了它的部分信息，使得积分的计算不可能. 要想精确地计算积分，需要恰当选取被积函数和 MC 权重函数；若想要提高计算效率 (积分收敛到精确值的速度)，则需要谨慎选取初始建议分布的形式，如本题中的 γ 选取.

6 源代码

6.1 FORTRAN90 代码

```
MODULE Metropolis
IMPLICIT NONE

CONTAINS
SUBROUTINE Sample1(a, b, gam, num)
    REAL(KIND=8) ,INTENT(IN) :: gam
    REAL(KIND=8) :: rand(num), integral
    REAL(KIND=8) xt, x(0:num), r, seed
    INTEGER(KIND=4), INTENT(IN) :: a, b, num
    INTEGER(KIND=4) :: i
    x(0) = 1
    integral = 0
    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(num, int(2147483647 * seed), rand)
    DO i = 1, num
        xt = - gam * LOG(rand(i))
        r = (xt / x(i-1))**(a - 1) * EXP(-(xt - x(i-1)/b)) * EXP
            ((xt - x(i-1))/gam)
        IF(rand(i) < MIN(1.0, r)) THEN
            x(i) = xt
```

```
21          ELSE
22              x(i) = x(i-1) ! 若建议未通过，仍保留结果与上一步相同
                    .
23          END IF
24          integral = real(integral * (i-1)) / i + (x(i) - a * b)
               **2 / i
25          ! 按步求平均值，以防止求和溢出
26      END DO
27      OPEN (1, ACCESS='append', file='integral.dat')
28      WRITE (1, *) integral
29      CLOSE(1)
30      print *, integral
31  END SUBROUTINE Sample1
32
33  SUBROUTINE Sample2(a, b, gam, num)
34      REAL(KIND=8) ,INTENT(IN) :: gam
35      REAL(KIND=8) :: rand(num), integral
36      REAL(KIND=8) xt, x(0:num), r, seed
37      INTEGER(KIND=4), INTENT(IN) :: a, b, num
38      INTEGER(KIND=4) :: i
39      x(0) = 1
40      integral = 0
41      CALL RANDOM_NUMBER(seed)
42      ! 用FORTRAN自带的随机数生成器生成16807生成器的种子
43      CALL Schrage(num, int(2147483647 * seed), rand)
44      DO i = 1, num
45          xt = - gam * LOG(rand(i))
46          r = (xt / x(i-1))**(a - 1) * (xt - a * b)**2 / (x(i-1) -
                a * b)**2&
47              * EXP(-(xt - x(i-1)/b)) * EXP((xt - x(i-1))/gam)
48          IF(rand(i) < MIN(1.0, r)) THEN
49              x(i) = xt
50          ELSE
51              x(i) = x(i-1) ! 若建议未通过，仍保留结果与上一步相同
                    .
52          END IF
53          integral = real(integral * (i-1)) / i + 1.0 / i
54          ! 直接求抽样结果x的平均值
55      END DO
```

```
56      OPEN (1, ACCESS='append', file='integral_2.dat')
57      WRITE (1, *) integral
58      CLOSE(1)
59      print *, integral
60  END SUBROUTINE Sample2
61  END MODULE Metropolis
62
63  SUBROUTINE Schrage(num, z0, rand)
64      !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
65      IMPLICIT NONE
66      INTEGER(KIND=4) :: N = 1, num
67      INTEGER :: m = 2147483647, a = 16807, q = 127773, r = 2836,
            In(num), z0
68      REAL(KIND=8), INTENT(INOUT) :: rand(num)
69      In(1) = z0 !将传入值z0作为种子
70      rand(1) = REAL(In(1))/m
71      DO N = 1, num - 1
72          In(N + 1) = a * MOD(In(N), q) - r * INT(In(N) / q)
73          IF (In(N + 1) < 0) THEN !若值小于零, 按Schrage方法加m
74              In(N + 1) = In(N + 1) + m
75          END IF
76          rand(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
77      END DO
78  END SUBROUTINE Schrage
79
80  PROGRAM MAIN
81      USE Metropolis
82      IMPLICIT NONE
83      INTEGER(KIND=4) :: i
84      REAL(KIND=8) :: gam
85      DO i = 1, 250
86          gam = 0.1 * i
87          CALL Sample1(3, 2, gam, 1000000)
88      END DO
89  END PROGRAM MAIN
```

6.2 python 绘图脚本代码

```python
import numpy as np
import matplotlib.pyplot as plt
import math
import matplotlib as mpl
mpl.use('TkAgg')

plt.rcParams['savefig.dpi'] = 300
plt.rcParams['figure.dpi'] = 300

#积分1
x = np.arange(0, 25, 0.1)
y = np.loadtxt('integral.dat')
y1 = np.abs(y - 12) / 12
plt.xlabel('$\gamma$')
plt.ylabel('Integral')
plt.plot(x, y)
y0 = 12 * np.ones(250)
plt.plot(x, y0)
plt.legend(['Computational value', 'Theroretical value'])
plt.savefig('integral1_1.eps')

plt.ylim(-5, 30)
plt.xlim(5, 25)
plt.savefig('integral1_2.eps')
plt.show()

plt.xlim(5, 25)
plt.ylim(-0.2, 1)
plt.plot(x, y1)
plt.xlabel('$\gamma$')
plt.ylabel('relative error')
plt.savefig('error1.eps')
plt.show()
```