

## 蒙特卡洛方法 作业 1

*October 10, 2023*

李永阳 2021141220025

**1****题 1.3**

我认为在此处使用  $\delta$  函数表示概率密度完全是多此一举, 因此使用分布表表示:

Table 1: 概率分布表

$x$	0	1	2
$P$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$

显然有:

$$\begin{aligned}\langle x \rangle &= \sum P_i \cdot x_i \\ \langle x^2 \rangle &= \sum P_i \cdot x_i^2 \\ \sigma^2 &= \langle x^2 \rangle - \langle x \rangle^2\end{aligned}$$

因此我们可以得到:

$$\begin{aligned}\langle x \rangle &= \frac{2}{3} \\ \langle x^2 \rangle &= 1 \\ \sigma^2 &= \frac{5}{9} \\ \sigma &= \frac{\sqrt{5}}{3}\end{aligned}$$

我们给出 CDF 图像 1a 如下。

**题 1.2**

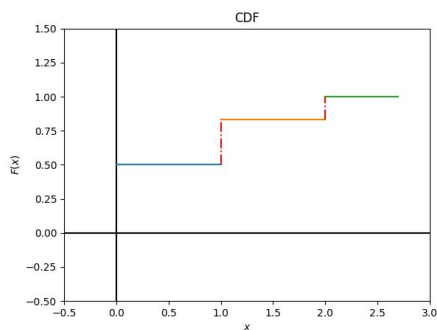
同上题:

Table 2: 概率分布表

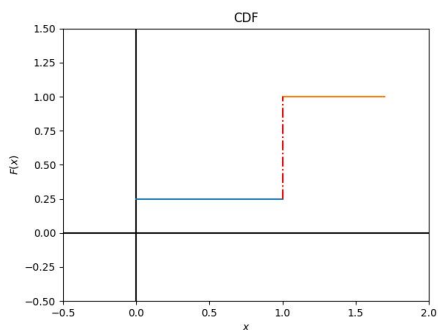
$x$	0	1
$P$	$\frac{1}{4}$	$\frac{3}{4}$

因此我们可以得到:

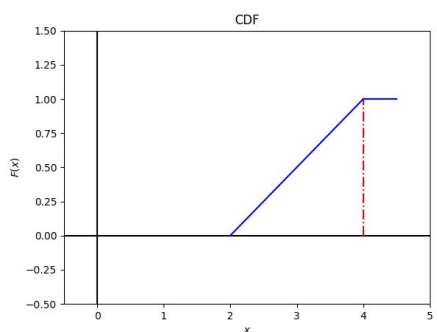
Figure 1: 概率分布函数



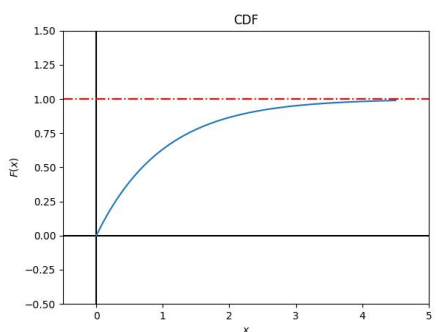
(a) 题 1.3 概率分布函数



(b) 题 1.2 概率分布函数



(c) 题 1.5 概率分布函数



(d) 题 1.6 概率分布函数

$$\begin{aligned}\langle x \rangle &= \frac{3}{4} \\ \langle x^2 \rangle &= \frac{3}{4} \\ \sigma^2 &= \frac{3}{16} \\ \sigma &= \frac{\sqrt{3}}{4}\end{aligned}$$

我们给出 CDF 图像 1b 如下。

## 题 1.5

$$\begin{aligned}\langle x \rangle &= \int_2^4 x f(x) dx = 3 \\ \langle x^2 \rangle &= \int_2^4 x^2 f(x) dx = \frac{28}{3} \\ \sigma^2 &= \frac{1}{3} \\ \sigma &= \frac{\sqrt{3}}{3}\end{aligned}$$

我们给出 CDF 图像 1c 如上。  
我们给出 PDF 图像 2a 如下。

## 题 1.6

$$\langle x \rangle = \int_0^{\infty} x e^{-x} dx = 1$$

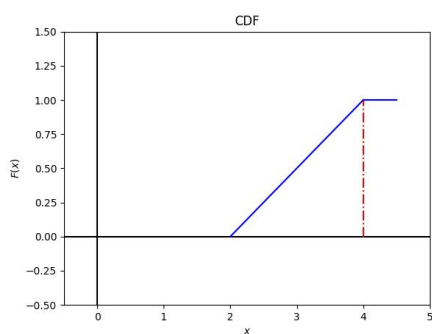
$$\langle x^2 \rangle = \int_0^{\infty} x^2 e^{-x} dx = 2$$

$$\sigma^2 = 1$$

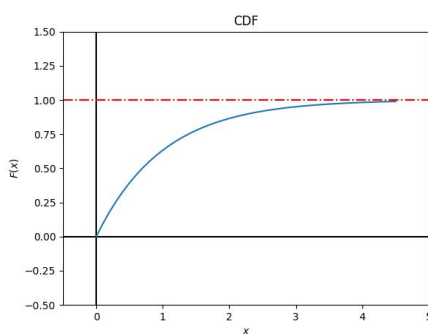
$$\sigma = 1$$

我们给出 CDF 图像 1d 如上。  
我们给出 PDF 图像 2b 如下。

Figure 2: 概率密度函数



(a) 题 1.5 概率密度函数



(b) 题 1.6 概率密度函数

## 2

### 题 2.1

非常显然，有：

$$y = \begin{cases} 0 & x \in [0, 0.5) \\ 1 & x \in [0.5, 5/6) \\ 2 & x \in [5/6, 1] \end{cases}$$

### 题 2.3

类似的，有

$$\begin{aligned} z = F(x) &= \int_{-\infty}^x f(t) dt \\ &= \int_{-h}^x \frac{dt}{2h} \\ &= \frac{x+h}{2h} \end{aligned}$$

求出其反函数：  $x = h(2z - 1)$

## 题 2.6

同样的，有：

$$\begin{aligned} z = F(x) &= \int_0^x f(t) dt \\ &= \int_0^x 2te^{-t^2} dt \\ &= 1 - e^{-x^2} \end{aligned}$$

求出其反函数： $x = \sqrt{-\ln(1-z)}$

## 题 2.7

同样的，有：

$$\begin{aligned} z = F(x) &= \int_0^x 3t^2 dt \\ &= x^3 \end{aligned}$$

求出其反函数： $x = z^{1/3}$

## 2D-numerical-solution

显然该问题在二维情况下转换成利用蒙特卡洛方法计算积分

$$\int_0^2 \int_0^2 3 - \frac{x}{2} - \frac{y}{2} dx dy$$

我们利用 numpy1.25.1 中自带的 0-1 随机数生成器 `numpy.random.random` 计算程序，numpy 版本号可以利用指令 `numpy.__version__` 查询

取样点数： $1e7$

计算结果： $8.000621379470351 \pm 0.00012908204728442695$

计算精度： $1e-3$

## 3D-numerical-solution

我不是很理解题中在 3D 绘景下计算具体指的是什么意思，不过我猜测是在  $[0, 2] \times [0, 2] \times [0, 3]$  中随机撒点，通过统计在题中所说平面之下的点的数目来估计体积。依然利用 numpy1.25.1 中自带的 0-1 随机数生成器 `numpy.random.random` 计算程序。

取样点数： $1e7$

计算结果： $8.001054 \pm 0.0017887365957195208$

计算精度： $1e-2$

## 十维蒙特卡洛积分

我们当然的意识到的，十维蒙特卡洛积分与二维蒙特卡洛积分并无太大区别。我们还可以猜想到，随着蒙特卡洛积分维度的增加，其精度的增加速度必然是在下降的。

取样点数为:  $1e6$

计算结果:  $0.28884213742394654 \pm 3.0800365850647203e - 06$

计算精度:  $1e - 2$

取样点数为:  $1e7$

计算结果:  $0.28430519116979497 \pm 9.449811116292106e - 07$

计算精度:  $1e - 2$

取样点数为:  $1e8$

计算结果:  $0.2847903425129942 \pm 3.0051152344585305e - 07$

计算精度:  $1e - 2$

## *References*

此次 homework 未用到任何参考文献

## Appendix

这是此次 homework 用到的全部代码

图像绘制代码:

```
1 import os
2 figure_save_path = "figure"
3 import warnings
4 warnings.filterwarnings("error")
5 import numpy as np
6 np.random.seed(0)
7 import matplotlib.pyplot as plt
8 from PIL import Image
9
10
11 ##plt.title("CDF")
12 ##plt.xlabel("$x$")
13 ##plt.ylabel("$F(x)$", rotation=90)
14 ##plt.xlim(-0.5, 2)
15 ##plt.ylim(-0.5, 1.5)
16 ##plt.axhline(0, c="black")
17 ##plt.axvline(0, c="black")
18 ##x1 = np.arange(0, 1.1, 0.1)
19 ##x2 = np.arange(1, 1.8, 0.1)
20 ##plt.plot(x1, 1/4*np.ones(x1.size))
21 ##plt.plot(x2, 1*np.ones(x2.size))
22 ##plt.plot([1, 1], [1/4, 1], "r-")
23 ##if not os.path.exists(figure_save_path):
24 ##    os.makedirs(figure_save_path)
25 ##plt.savefig(os.path.join(figure_save_path, "1-2-CDF" + ".jpg"))
26
27 ##plt.title("CDF")
28 ##plt.xlabel("$x$")
29 ##plt.ylabel("$F(x)$", rotation=90)
30 ##plt.xlim(-0.5, 3)
31 ##plt.ylim(-0.5, 1.5)
32 ##plt.axhline(0, c="black")
33 ##plt.axvline(0, c="black")
34 ##x1 = np.arange(0.0, 1.1, 0.1)
35 ##x2 = np.arange(1.0, 2.1, 0.1)
36 ##x3 = np.arange(2.0, 2.8, 0.1)
37 ##plt.plot(x1, 1/2*np.ones(x1.size))
38 ##plt.plot(x2, 5/6*np.ones(x2.size))
39 ##plt.plot(x3, 1*np.ones(x3.size))
40 ##plt.plot([1, 1], [1/2, 5/6], "r-")
41 ##plt.plot([2, 2], [5/6, 1], "r-")
42 ##if not os.path.exists(figure_save_path):
43 ##    os.makedirs(figure_save_path)
44 ##plt.savefig(os.path.join(figure_save_path, "1-3-CDF" + ".jpg"))
45
46 ##plt.title("PDF")
47 ##plt.xlabel("$x$")
48 ##plt.ylabel("$P(x)$")
49 ##plt.xlim(1.5, 4.5)
50 ##plt.ylim(0.0, 0.8)
51 ##plt.axhline(0, c="black")
52 ##plt.axvline(0, c="black")
53 ##x1 = np.arange(2, 4.1, 0.1)
54 ##plt.plot(x1, 1/2*np.ones(x1.size))
```

```

55  ##plt.plot([2, 2], [0, 0.5], "r-.")
56  ##plt.plot([4, 4], [0, 0.5], "r-.")
57  ##if not os.path.exists(figure_save_path):
58  ##    os.makedirs(figure_save_path)
59  ##plt.savefig(os.path.join(figure_save_path, "1-5-PDF" + ".jpg"))
60
61  ##plt.title("CDF")
62  ##plt.xlabel("$x$")
63  ##plt.ylabel("$F(x)$")
64  ##plt.xlim(-0.5, 5.0)
65  ##plt.ylim(-0.5, 1.5)
66  ##plt.axhline(0, c="black")
67  ##plt.axvline(0, c="black")
68  ##x1 = np.arange(2, 4.1, 0.1)
69  ##f = lambda x:0.5*(x-2)
70  ##plt.plot(x1, f(x1), c="b")
71  ##plt.plot([4, 4], [0, 1], "r-.")
72  ##plt.plot([4, 4.5], [1, 1], c="b")
73  ##if not os.path.exists(figure_save_path):
74  ##    os.makedirs(figure_save_path)
75  ##plt.savefig(os.path.join(figure_save_path, "1-5-CDF" + ".jpg"))
76
77  ##plt.title("CDF")
78  ##plt.xlabel("$x$")
79  ##plt.ylabel("$F(x)$")
80  ##plt.xlim(-0.5, 5.0)
81  ##plt.ylim(-0.5, 1.5)
82  ##plt.axhline(0, c="black")
83  ##plt.axvline(0, c="black")
84  ##x1 = np.arange(0, 4.6, 0.1)
85  ##f = lambda x:1-np.exp(-x)
86  ##plt.plot(x1, f(x1))
87  ##plt.axhline(1, c="r", linestyle="-.")
88  ##if not os.path.exists(figure_save_path):
89  ##    os.makedirs(figure_save_path)
90  ##plt.savefig(os.path.join(figure_save_path, "1-6-CDF" + ".jpg"))
91
92  plt.title("PDF")
93  plt.xlabel("$x$")
94  plt.ylabel("$P(x)$")
95  plt.xlim(-0.5, 4.5)
96  plt.ylim(-0.2, 1.2)
97  plt.axhline(0, c="black")
98  plt.axvline(0, c="black")
99  f = lambda x:np.exp(-x)
100 x1 = np.arange(0, 4.1, 0.1)
101 plt.plot(x1, f(x1))
102 if not os.path.exists(figure_save_path):
103 os.makedirs(figure_save_path)
104 plt.savefig(os.path.join(figure_save_path, "1-6-PDF" + ".jpg"))

```

题五代码一：蒙特卡洛方法二维积分

```

1  import os
2  figure_save_path = "file_fig"
3  import warnings
4  warnings.filterwarnings("error")
5  import numpy as np
6  np.random.seed(0)
7  import matplotlib.pyplot as plt

```

```
8 from PIL import Image
9
10 print("RNG:", "Python numpy.random.random")
11
12 X_start = 0
13 Y_start = 0
14
15 X_end = 2
16 Y_end = 2
17
18 N = 10000000
19
20 num_points = list(range(N))
21 random_xpoints = np.random.uniform(X_start, X_end, N)
22 random_ypoints = np.random.uniform(Y_start, Y_end, N)
23
24 X = np.arange(X_start, X_end, 100)
25 Y = np.arange(Y_start, Y_end, 100)
26
27 f = lambda i: 3 - random_xpoints[i]/2 - random_ypoints[i]/2
28 guess = np.array(list(map(f, num_points)))
29 ave = sum(guess)/N
30 I = ave*(X_end-X_start)*(Y_end-Y_start)
31
32 S2 = sum((guess-ave)**2)/(N-1)
33 S = S2**0.5
34
35 print(I, "\pm", S/N**0.5)
36
37 ##running result
38 ##RNG: Python numpy.random.random
39 ##8.000621379470351 \pm 0.00012908204728442695
```

题五代码二：蒙特卡洛方法二维点估计

```
1 import os
2 figure_save_path = "file_fig"
3 import warnings
4 warnings.filterwarnings("error")
5 import numpy as np
6 np.random.seed(0)
7 import matplotlib.pyplot as plt
8 from PIL import Image
9
10 print("RNG:", "Python numpy.random.random")
11 N = 10000000
12
13 num = list(range(N))
14 xpoints = np.random.random(N) * 2
15 ypoints = np.random.random(N) * 2
16 zpoints = np.random.random(N) * 3
17
18 V = 2 * 2 * 3
19
20 f = lambda i: zpoints[i] <= 3 - xpoints[i]/2 - ypoints[i]/2
21
22 counts = np.sum(np.array(list(map(f, num))))
23 ave = counts/N
24 S2 = counts*(1-ave)**2/(N-1) + (N-counts)*ave**2/(N-1)
25 S = S2**0.5
```



```
26 print(ave*V, "\pm", S/N**0.5*V)
```

题七代码:

```
1 import os
2 figure_save_path = "figure"
3 import warnings
4 warnings.filterwarnings("error")
5 import numpy as np
6 np.random.seed(0)
7 import matplotlib.pyplot as plt
8 from PIL import Image
9
10 X1_start = 0
11 X2_start = 0
12 X3_start = 0
13 X4_start = 0
14 X5_start = 0
15
16 X6_start = 0
17 X7_start = 0
18 X8_start = 0
19 X9_start = 0
20 X0_start = 0
21
22
23 X1_end = 2
24 X2_end = 2
25 X3_end = 2
26 X4_end = 2
27 X5_end = 2
28
29 X6_end = 2
30 X7_end = 2
31 X8_end = 2
32 X9_end = 2
33 X0_end = 2
34
35
36 N = 100000000
37
38 num_points = list(range(N))
39 random_x1points = np.random.uniform(X1_start, X1_end, N)
40 random_x2points = np.random.uniform(X2_start, X2_end, N)
41 random_x3points = np.random.uniform(X3_start, X3_end, N)
42 random_x4points = np.random.uniform(X4_start, X4_end, N)
43 random_x5points = np.random.uniform(X5_start, X5_end, N)
44
45 random_x6points = np.random.uniform(X6_start, X6_end, N)
46 random_x7points = np.random.uniform(X7_start, X7_end, N)
47 random_x8points = np.random.uniform(X8_start, X8_end, N)
48 random_x9points = np.random.uniform(X9_start, X9_end, N)
49 random_x0points = np.random.uniform(X0_start, X0_end, N)
50
51 f = lambda i: np.exp(-(random_x1points[i]**2 +
52 random_x2points[i]**2 +
53 random_x3points[i]**2 +
54 random_x4points[i]**2 +
55 random_x5points[i]**2 +
56
```

```
57 random_x6points[i]**2 +
58 random_x7points[i]**2 +
59 random_x8points[i]**2 +
60 random_x9points[i]**2 +
61 random_x0points[i]**2 ))
62
63 guess = np.array(list(map(f, num_points)))
64 ave = sum(guess)/N
65 I = ave*(X1_end-X1_start)*\
66 (X2_end-X2_start)*\
67 (X3_end-X3_start)*\
68 (X4_end-X4_start)*\
69 (X5_end-X5_start)*\
70 (X6_end-X6_start)*\
71 (X7_end-X7_start)*\
72 (X8_end-X8_start)*\
73 (X9_end-X9_start)*\
74 (X0_end-X0_start)
75
76 S2 = sum((guess-ave)**2)/(N-1)
77 S = S2**0.5
78
79 print(I, "\pm", S/N**0.5)
```