Assignment 6

12310520

芮堤涵

T1. (1) Let X be the number of heads in n tosses, Y be the number of tails in n tosses.

$X+Y=n \quad Var(X)=Var(Y)=n\cdot\frac{1}{2}\times(1-\frac{1}{2})=\frac{1}{4}n$

$Cov(X,Y)=Cov(X,n-X)=Cov(X,n)-Cov(X,X)=-Var(X)$

$Var(X-Y)=Var(X)+Var(Y)-2Cov(X,Y)$

$=\frac{1}{4}n+\frac{1}{4}n-2\times(-\frac{1}{4}n)=n$

(2) No, I don't agree with Bob.

$Var(X-Y)=n$, the standard deviation is $\sqrt{n}$.

As n increases, the range of fluctuation in $X-Y$ will grow larger. Therefore, $X-Y$ will not converge to 0

T2. (1) $X_i \sim Poisson(3)$ $E(X_i)=3$ $Var(X_i)=3$

$E(Y_n)=\frac{E(X_1)+E(X_2)+\cdots+E(X_n)}{n}=3$

$Var(Y_n)=\frac{Var(X_1)+Var(X_2)+\cdots+Var(X_n)}{n^2}=\frac{n\times 3}{n^2}=\frac{3}{n}$

for CLT, $Y_n \overset{approx}{\sim} N(3,\frac{3}{n})$,

as $n\to\infty$, $Var(Y_n)\to 0$, so that Y_n converge in probability to 3.

(2) $X_i \sim U(-1,3)$ $E(X_i)=\frac{-1+3}{2}=1$ $Var(X_i)=\frac{(3+1)^2}{12}=\frac{4}{3}$

$E(Y_n)=\frac{1\cdot n}{n}=1$ $Var(Y_n)=\frac{n\times\frac{4}{3}}{n^2}=\frac{4}{3n}$

for CLT, $Y_n \underbrace{\sim}_{}^{approx} N(1,\frac{4}{3n})$,

as $n\to\infty$, $Var(Y_n)\to 0$, so that Y_n converge in probability to 1

(3) $X_i \sim exp(5)$ $E(X_i)=\frac{1}{5}$ $Var(X_i)=\frac{1}{25}$

$E(Y_n)=\frac{n}{5n}=\frac{1}{5}$ $Var(Y_n)=\frac{1}{25n}$

for CLT, $Y_n \overset{approx}{\sim} N(\frac{1}{5},\frac{1}{25n})$, as $n\to\infty$, $Var(Y_n)\to 0$

so that Y_n converge in probability to $\frac{1}{5}$

T3. let $R=\sqrt{-2\ln(U_1)}$ $U_1\sim U[0,1]$

$\theta=2\pi U_2$ $U_2\sim U[0,1]\Rightarrow\theta\sim U[0,2\pi]$

$f_\theta(\theta)=\frac{1}{2\pi}$

$F_R(r)=P(R\le r)=P(-2\ln(U_1)\le r^2)$

$=P(U_1\ge e^{-\frac{r^2}{2}})=1-e^{-\frac{r^2}{2}}$

$f_R(r)=\frac{d}{dr}F_R(r)=re^{-\frac{r^2}{2}}$

$Z_1=r\cos\theta$ $Z_2=r\sin\theta$

$J=\begin{vmatrix}\frac{\partial Z_1}{\partial R} & \frac{\partial Z_1}{\partial\theta}\\ \frac{\partial Z_2}{\partial R} & \frac{\partial Z_2}{\partial\theta}\end{vmatrix}=\begin{vmatrix}\cos\theta & -r\sin\theta\\ \sin\theta & r\cos\theta\end{vmatrix}=r$ $\Rightarrow dz_1dz_2=r\,dr\,d\theta$

$P(Z_1\le a, Z_2\le b)=\int_{-\infty}^{a}\int_{-\infty}^{b}f(z_1,z_2)\,dz_2\,dz_1$

$=\iint f(z_1,z_2)\,r\,d\theta\,dr$

$$= \int\int \frac{f(r,\theta)}{r} \, r \, d\theta \, dr$$

$$= \int\int f(r,\theta) \, d\theta \, dr = \int\int \frac{r e^{-\frac{r^2}{2}}}{2\pi} \, dr \, d\theta$$

$$= \int_{-\infty}^{a} \int_{-\infty}^{b} \frac{e^{-\frac{z_1^2+z_2^2}{2}}}{2\pi} \, dz_2 \, dz_1$$

$$= \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}} e^{-\frac{z_1^2}{2}} dz_1 \cdot \int_{-\infty}^{b} \frac{1}{\sqrt{2\pi}} e^{-\frac{z_1^2}{2}} dz_2$$

$$= \Phi(a) \, \Phi(b)$$

T4. (1) $X \sim \text{Geometric}(p)$

$p_k = P(X=k) = p(1-p)^{k-1} \quad k=1,2,\cdots$

divide the interval $(0,1)$ into subintervals

$I_1 = (0, p_1)$, $I_2 = [p_1, p_1+p_2)$, $I_3 = [p_1+p_2, p_1+p_2+p_3)$ $\cdots$

define $x_i = k$ if $u_i \in I_k$, then $x_1, x_2 \cdots x_n$ can be considered numbers generated from Geometric (p).

(2) $f(x) = \frac{1}{\pi(1+x^2)} \quad x \in (-\infty, \infty)$ 有 $f(x)$ 求 $F(x)$

$F_X(x) = \int_{-\infty}^{x} \frac{1}{\pi(1+t^2)} dt = \frac{1}{\pi} \tan^{-1} x + C$ $\Rightarrow x = F^{-1}(u)$ 对每个 $u \sim U(0,1)$ 变换

for symmetry, $F_X(0) = \frac{1}{2} \Rightarrow C = \frac{1}{2}$, $F_X(x) = \frac{1}{\pi} \tan^{-1} x + \frac{1}{2}$

$F^{-1}(x) = \tan(\pi(x - \frac{1}{2}))$ let $U \sim U[0,1]$

then $X = F^{-1}(U)$

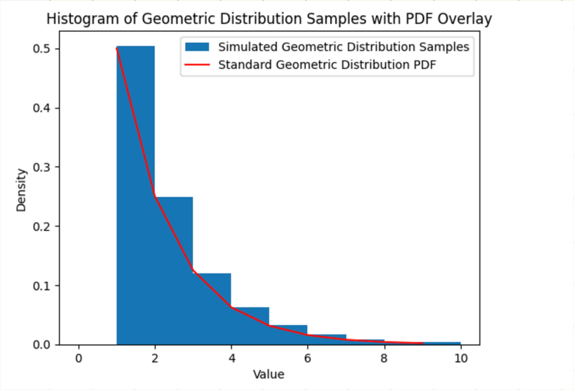
therefore, for $u_1, u_2, \cdots u_n$ from a uniform distribution random number generator, define $x_i = \tan(\pi(u_i - \frac{1}{2}))$

then $x_1, x_2, \cdots x_n$ can be considered numbers generated from $f(x) = \frac{1}{\pi(x^2+1)}$

T5

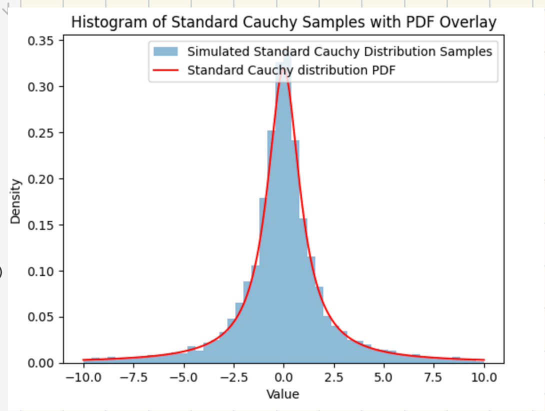
(1)

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import geom
np.random.seed(20241128)

n_samples = 10000

geo_samples=np.random.geometric(0.5,n_samples)


plt.hist(geo_samples,
label='Simulated Geometric Distribution Samples',
density=True,range=(0,10))

x = range(1,10,1);
pdf = geom.pmf(x,p=0.5)
plt.plot(x, pdf, 'r-', label=f'Standard Geometric Distribution PDF')

# Add labels and legend
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram of Geometric Distribution Samples with PDF Overlay')
plt.legend()

# Show the plot
plt.show()
```



(2)

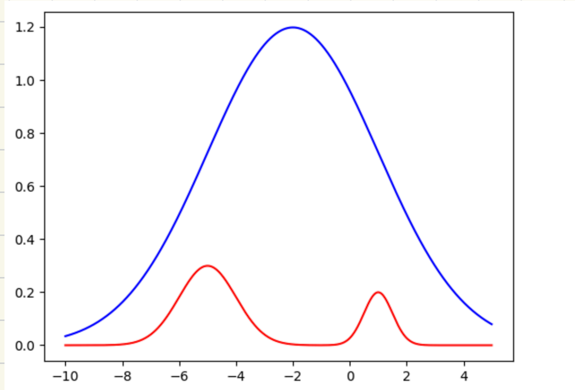
```python
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(20241128)

n_samples = 10000

# Step 1: Generate uniform samples
unif_samples = np.random.uniform(0, 1, n_samples)

# Step 2: Transform using the inverse of the exponential CDF
cauchy_samples = np.tan(np.pi*(unif_samples-0.5))

# Step 3: Plotting
# Create a histogram of the Standard Cauchy distribution samples
plt.hist(cauchy_samples, bins=50, alpha=0.5,
label='Simulated Standard Cauchy Distribution Samples', density=True,range=(-10,10))

# Overlay the theoretical PDF of the Standard Cauchy distribution
def cauchy(x):
    return 1/(np.pi*(1+x*x))
x = np.linspace(-10,10,1000);
pdf = cauchy(x)
plt.plot(x, pdf, 'r-', label=f'Standard Cauchy distribution PDF')

# Add labels and legend
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Histogram of Standard Cauchy Samples with PDF Overlay')
plt.legend()

# Show the plot
plt.show()
```



T6. (1) theoretical $f(x)=\frac{5}{4\sqrt{2\pi}}\left(0.6\exp\left(-\frac{(x+5)^2}{2}\right)+0.4\exp\left(-\frac{(x-1)^2}{0.5}\right)\right)$

proposal distribution $g(x)=\frac{1}{\sqrt{2\pi}\times 3}e^{-\frac{x+2}{18}}$

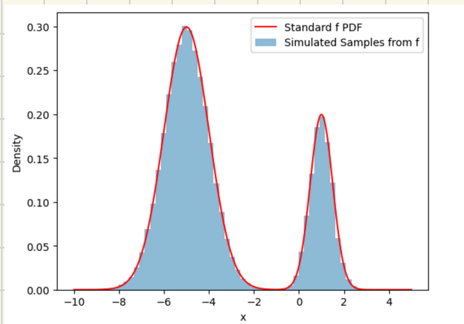
$c=9$



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon
np.random.seed(20241128)
def f(x):
    return 5/(4*np.sqrt(2*np.pi))*(0.6 * np.exp((-(x+5) ** 2)/2) + 0.4 * np.exp(-(x-1) ** 2 / 0.5))

x = np.arange(-10, 5, 0.01)

plt.plot(x, f(x), color = 'red')

# Try to use a normal distribution to be the proposal distribution
def g(x, mu, sigma, c):
    return c / np.sqrt(2 * np.pi * sigma**2) * np.exp(- (x - mu)**2 / (2 * sigma**2))

plt.plot(x, g(x, -2, 3, 9),color='blue') # by trial and error
```

(2)

```python
# Apply the rejection sampling technique
n_samples = 500000
# Step I: sample from the proposal distribution
x_samples = np.random.normal(loc=-2, scale=3, size=n_samples)

# Step II: sample from the uniform distribution
u_samples = np.random.uniform(0, 1, n_samples)

# Step III: compare ui with f(xi)/cg(xi) and decide whether to reject or not
accept_samples = x_samples[u_samples <= f(x_samples) / g(x_samples, -2, 3, 9)]
x=np.linspace(-10,5,1000);

plt.plot(x,f(x),'r-',label='Standard f PDF')
plt.hist(accept_samples, bins = 50, alpha=0.5, label='Simulated Samples from f', density=True)

plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.show()
```



(3)

```python
acceptance_proportion = len(accept_samples) / (n_samples) * 100

print(f"Acceptance proportion = {acceptance_proportion:.2f} %")
```

```
Acceptance proportion = 11.08 %
```

acceptance _ proportion $=\frac{1}{c}=\frac{1}{9}$

T7. (1) for the CLT, $\hat{p}_n \overset{\text{approx}}{\sim} N(p, \frac{p(1-p)}{n})$

$$\Rightarrow \frac{\sqrt{n}(\hat{p}_n - p)}{\sqrt{p(1-p)}} \overset{\text{approx}}{\sim} N(0,1)$$

$$\Rightarrow P(|\hat{p}_n - p| \leq 0.005) = P\left(-\frac{0.005\sqrt{n}}{\sqrt{p(1-p)}} \leq \frac{\sqrt{n}(\hat{p}_n - p)}{\sqrt{p(1-p)}} \leq \frac{0.005\sqrt{n}}{\sqrt{p(1-p)}}\right)$$

$$\approx 2\Phi\left(\frac{0.005\sqrt{n}}{\sqrt{p(1-p)}}\right) - 1$$

$$\Rightarrow 2\Phi\left(\frac{0.005\sqrt{n}}{\sqrt{p(1-p)}}\right) - 1 \geq 0.95 \Rightarrow \Phi\left(\frac{0.005\sqrt{n}}{\sqrt{p(1-p)}}\right) \geq 0.975$$

$$\frac{0.005\sqrt{n}}{\sqrt{p(1-p)}} \geq 1.96 \quad \Rightarrow n \geq p(1-p) \times 153664$$

$p(1-p) \leq 0.25$ use $n \geq 0.25 \times 153664 \approx 38416$

let $n = 40000$

```python
import numpy as np

m = 20
n = 50
sample_time = 40000
count = 0

a = np.zeros((m, n))

for k in range(sample_time):
    a.fill(0)
    a[0][0] = 1
    total_sum = 0

    for i in range(m):
        for j in range(n):
            if j > 0:
                if a[i][j-1] == 1 and np.random.uniform(0, 1) <= 0.8:
                    a[i][j] = 1
            if i > 0:
                if a[i-1][j] == 1 and np.random.uniform(0, 1) <= 0.3:
                    a[i][j] = 1
            if a[i][j] == 1:
                total_sum += 1

    if total_sum > m * n * 0.3:
        count += 1

probability = count / sample_time * 100
print(f"Probability = {probability:.2f} %")
```

```
Probability = 0.31 %
```

$p \approx 0.31\%$

(2)

```python
import numpy as np

m = 20
n = 50
sample_time = 40000
X=0
XX=0

a = np.zeros((m, n))

for k in range(sample_time):
    a.fill(0)
    a[0][0] = 1
    total_sum = 0

    for i in range(m):
        for j in range(n):
            if j > 0:
                if a[i][j-1] == 1 and np.random.uniform(0, 1) <= 0.8:
                    a[i][j] = 1
            if i > 0:
                if a[i-1][j] == 1 and np.random.uniform(0, 1) <= 0.3:
                    a[i][j] = 1
            if a[i][j] == 1:
                total_sum += 1
    X=X+total_sum
    XX=XX+total_sum**2

average = X / sample_time
print(f"Average = {average:.4f} ")
sd =np.sqrt(XX/sample_time-average**2)
print(f"standard deviation ={sd:.4f}")
```

```
Average = 43.7098
standard deviation =63.6516
```

(3) $\sigma \approx 63.6516$

$E(X) \approx 43.7098$