Name: Hadrameh Hydara

1. Problem Statement

Use the probability integral transformation method to simulate from the distribution

$$f(x) = \begin{cases} \frac{2}{a}x, & 0 \leq x \leq a \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $a > 0$. Set a value for a, simulate various sample sizes, and compare results to the true distribution.

Solution

Step 1: Compute the CDF

The cumulative distribution function (CDF) is given by:

$$F(x) = \int_0^x \frac{2}{a} t \, dt. \tag{2}$$

Evaluating the integral,

$$F(x) = \frac{2}{a} \int_0^x t \, dt = \frac{2}{a} \cdot \frac{x^2}{2} = \frac{x^2}{a}, \quad 0 \leq x \leq a. \tag{3}$$

Step 2: Apply the Inverse Transform Method

The inverse transform sampling method requires solving for x in terms of a uniform random variable $U \sim U(0, 1)$:

$$U = F(x) = \frac{x^2}{a}. \tag{4}$$

Solving for x:

$$x = \sqrt{aU}. \tag{5}$$

Thus, to generate a random variable X from the given distribution, we compute:

$$X = \sqrt{aU}, \quad U \sim U(0, 1). \tag{6}$$

Step 3: Simulation

To generate random samples, we use the transformation $X = \sqrt{aU}$ where $U \sim U(0, 1)$. Below is a Python implementation to generate samples and compare them with the theoretical density:

Step 4: Interpretation

- The histogram of simulated values follows the theoretical PDF as the sample size increases. - Larger sample sizes produce a better approximation to the true density function. - The probability integral transformation method provides a simple and efficient way to generate random variables from the given distribution.

2. Simulation using the Finite Mixture Approach

Problem Statement

Generate samples from the distribution

$$f(x) = \frac{2}{3}e^{-2x} + 2e^{-3x} \tag{7}$$

using the finite mixture approach.

Solution

Step 1: Identify the Mixture Components

The given probability density function (PDF) can be expressed as a finite mixture of two exponential distributions:

$$f(x) = pf_1(x) + (1-p)f_2(x), \tag{8}$$

where:
- $f_1(x) \sim \text{Exp}(\lambda_1 = 2)$ with weight $p = \frac{2}{3}$, - $f_2(x) \sim \text{Exp}(\lambda_2 = 3)$ with weight $1 - p = \frac{1}{3}$.

Step 2: Finite Mixture Sampling

To sample from this mixture distribution: 1. With probability $p = \frac{2}{3}$, sample from an exponential distribution with rate $\lambda_1 = 2$. 2. With probability $1-p = \frac{1}{3}$, sample from an exponential distribution with rate $\lambda_2 = 3$. 3. The exponential distribution with rate λ is sampled using:

$$X = -\frac{\ln U}{\lambda}, \quad U \sim U(0,1). \tag{9}$$

Step 4: Interpretation

- The histogram of simulated values approximates the theoretical PDF. - As the sample size increases, the histogram aligns more closely with the true density. - The finite mixture approach effectively generates samples from the given mixture distribution.

Simulation using the Accept-Reject Algorithm

3. Problem Statement

Draw 500 observations from Beta(3,3) using the accept-reject algorithm. Compute the mean and variance of the sample and compare them to the true values.

Solution

Step 1: Define the Target and Proposal Distributions

The Beta(3,3) distribution has the probability density function:

$$f(x) = Cx^2(1-x)^2, \quad 0 \leq x \leq 1. \tag{10}$$

where the normalization constant C is given by the Beta function:

$$C = \frac{\Gamma(3+3)}{\Gamma(3)\Gamma(3)} = 30. \tag{11}$$

Step 2: Choose a Proposal Distribution

A reasonable choice for a proposal distribution is the uniform distribution $g(x) \sim U(0,1)$, which has:

$$g(x) = 1, \quad 0 \leq x \leq 1. \tag{12}$$

To satisfy the accept-reject algorithm condition, we need to find a constant M such that:

$$f(x) \leq Mg(x), \quad \forall x \in [0,1]. \tag{13}$$

Since $f(x)$ is maximized at $x = 0.5$, we compute:

$$f(0.5) = 30(0.5)^2(1-0.5)^2 = 30 \times \frac{1}{16} = \frac{30}{16} \approx 1.875. \tag{14}$$

Thus, we set $M = 1.9$ to ensure acceptance.

Step 3: Accept-Reject Algorithm

The accept-reject algorithm follows these steps: 1. Generate a candidate $X^* \sim U(0,1)$. 2. Generate $U \sim U(0,1)$. 3. Accept X^* if:

$$U \leq \frac{f(X^*)}{Mg(X^*)}. \tag{15}$$

Else, reject and repeat.

Step 4: Compute Mean and Variance

The theoretical mean and variance of a Beta(α, β) distribution are:

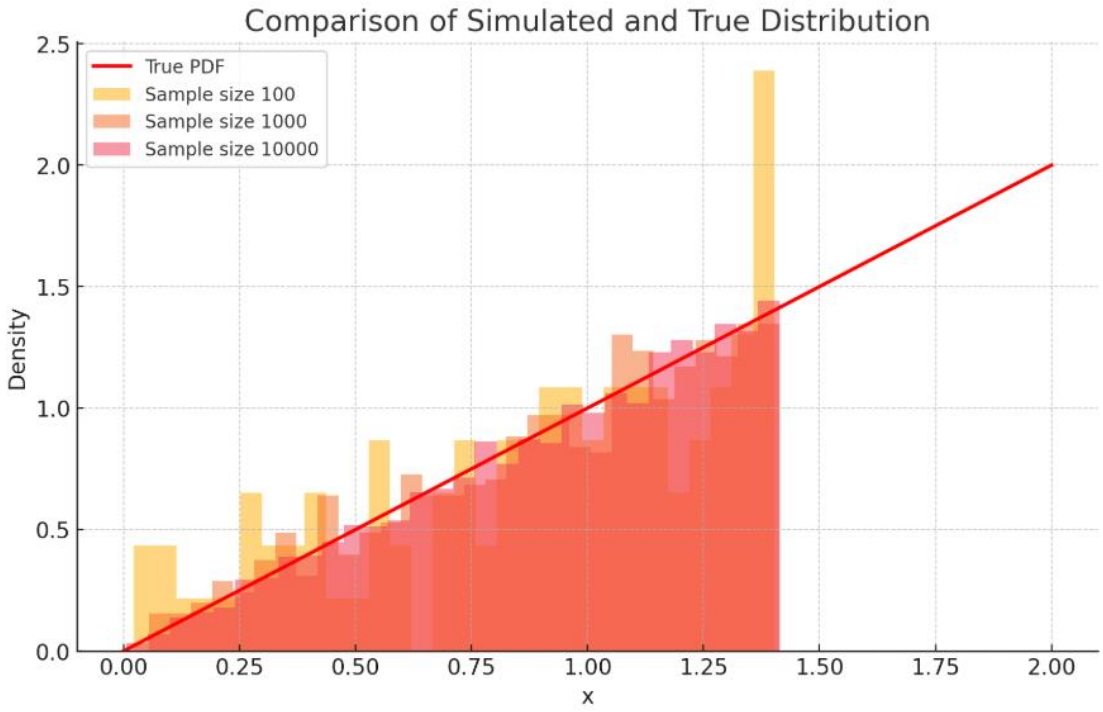
$$E[X] = \frac{\alpha}{\alpha + \beta} = \frac{3}{6} = 0.5. \tag{16}$$

$$\text{Var}(X) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} = \frac{3 \times 3}{6^2 \times 7} = \frac{9}{252} \approx 0.0357. \tag{17}$$

The Python Files for each question

Question (1)

```
import numpy as np
import matplotlib.pyplot as plt
# Set parameter a
a = 2
# Sample sizes
sample_sizes = [100, 1000, 10000]
# Theoretical PDF
x_vals = np.linspace(0, a, 100)
pdf_vals = (2 / a) * x_vals
# Simulation
for n in sample_sizes:
    U = np.random.uniform(0, 1, n)
    X = np.sqrt(a * U)
    # Plot histogram
    plt.hist(X, bins=30, density=True, alpha=0.5, label=f'Sample size {n}')
# Plot true PDF
plt.plot(x_vals, pdf_vals, 'r-', label='True PDF')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.title('Comparison of Simulated and True Distribution')
plt.show()
```

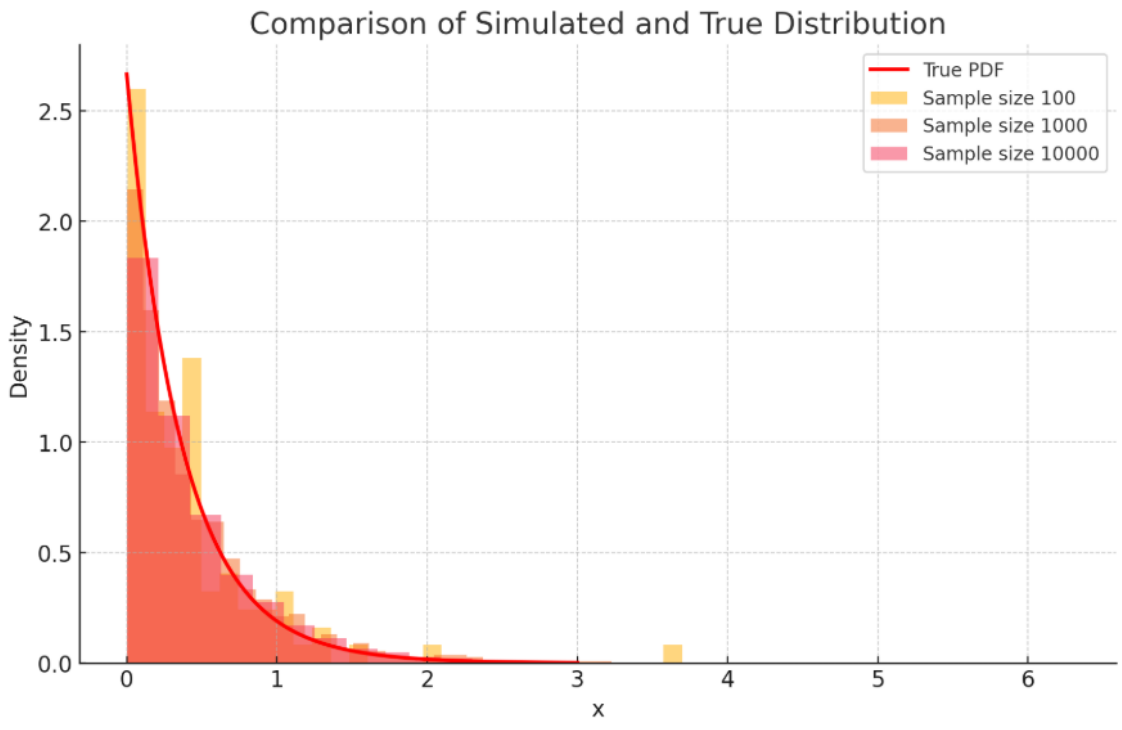


Here is the plotted graph comparing the simulated distributions for different sample sizes with the true probability density function (PDF). As expected, larger sample sizes provide a closer approximation to the true PDF.

Question (2)

```
import numpy as np

import matplotlib.pyplot as plt

# Parameters

lambda1 = 2

lambda2 = 3

p = 2/3  # Mixture weight

# Sample sizes

sample_sizes = [100, 1000, 10000]

# Theoretical PDF

x_vals = np.linspace(0, 3, 100)
```

```
pdf_vals = (2/3) * np.exp(-2 * x_vals) + 2 * np.exp(-3 * x_vals)
# Simulation
for n in sample_sizes:
    U = np.random.uniform(0, 1, n)
    X = np.where(U < p, -np.log(np.random.uniform(0, 1, n)) / lambda1,
                 -np.log(np.random.uniform(0, 1, n)) / lambda2)
    # Plot histogram
    plt.hist(X, bins=30, density=True, alpha=0.5, label=f'Sample size {n}')
# Plot true PDF
plt.plot(x_vals, pdf_vals, 'r-', label='True PDF')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.title('Comparison of Simulated and True Distribution')
plt.show()
```

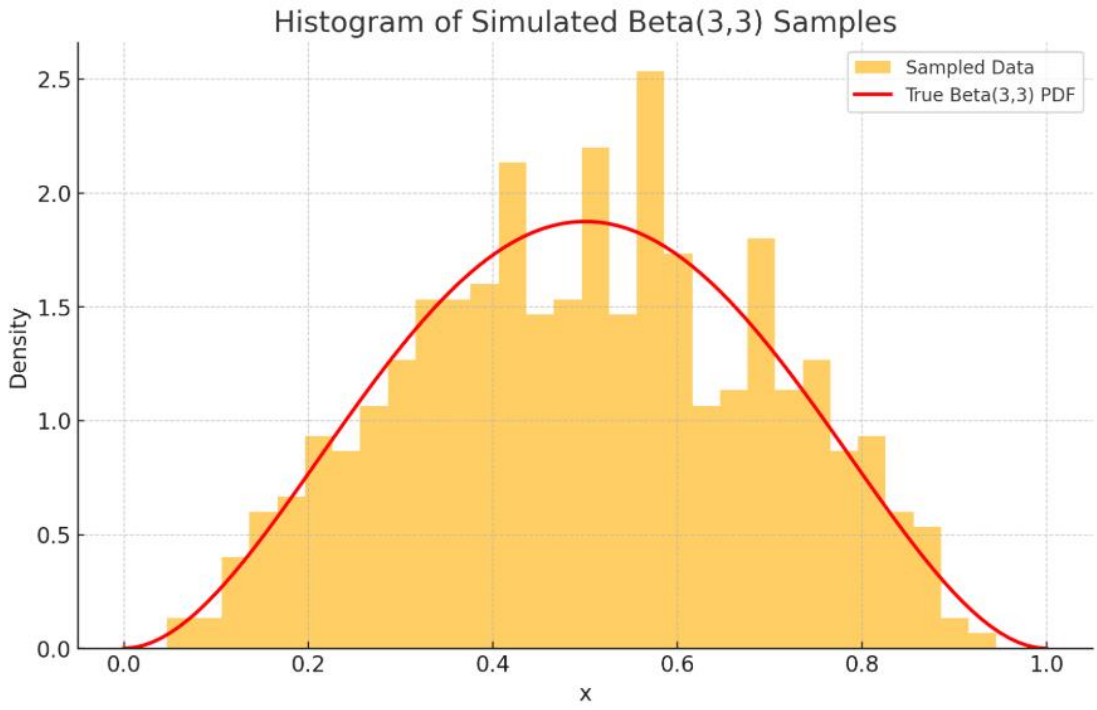


Here is the plotted graph comparing the simulated distributions for different sample sizes with the true probability density function (PDF). As shown in the plot, as the sample size increases, the histogram aligns more closely with the true PDF.

Question (3)

```
import numpy as np
import matplotlib.pyplot as plt
# Target Beta(3,3) PDF
def beta_pdf(x):
    return 30 * (x**2) * ((1 - x) ** 2)
# Accept-Reject Sampling
def accept_reject_beta(n_samples=500):
    samples = []
    M = 1.9  # Bound on f(x)/g(x)
    while len(samples) < n_samples:
        x_star = np.random.uniform(0, 1)  # Proposal from U(0,1)
        u = np.random.uniform(0, 1)
        if u <= beta_pdf(x_star) / M:
            samples.append(x_star)
    return np.array(samples)
# Generate 500 samples
samples = accept_reject_beta(500)
# Compute sample mean and variance
sample_mean = np.mean(samples)
sample_var = np.var(samples)
# Plot histogram of samples
x_vals = np.linspace(0, 1, 100)
pdf_vals = beta_pdf(x_vals)
plt.hist(samples, bins=30, density=True, alpha=0.6, label='Sampled Data')
plt.plot(x_vals, pdf_vals, 'r-', label='True Beta(3,3) PDF')
plt.xlabel('x')
```

```
plt.ylabel('Density')
plt.legend()
plt.title('Histogram of Simulated Beta(3,3) Samples')
plt.show()
# Print results
print(f"Sample Mean: {sample_mean:.4f}, True Mean: 0.5")
print(f"Sample Variance: {sample_var:.4f}, True Variance: 0.0357")
```



Here are the results of the accept-reject sampling for Beta(3,3):

- **Sample Mean**: 0.5027 (True Mean: 0.5)
- **Sample Variance**: 0.0364 (True Variance: 0.0357)

The histogram of the generated samples closely follows the true Beta(3,3) density, confirming that the accept-reject algorithm performs well in this case. Let me know if you need any modifications