Your Name: ABHISHEK VELUSAMY **Problem Set 2, Due: Mar. 18, 2025**

Problem 1: Probability Integral Transformation Method

The cumulative distribution function (CDF) of the given probability density function (PDF) is:

$$F(x) = \int_0^x \frac{2}{a^2} t dt = \frac{x^2}{a^2}, \quad 0 \leq x \leq a. \tag{1}$$

By inverting the CDF, we get:

$$x = a\sqrt{U}, \tag{2}$$

where $U \sim \text{Uniform}(0, 1)$. The following Python code generates samples:

```
import numpy as np
import matplotlib.pyplot as plt

def sample_pit(n, a=2):
    U = np.random.uniform(0, 1, n)
    X = a * np.sqrt(U)
    return X

samples = sample_pit(10000, a=2)
plt.hist(samples, bins=50, density=True, alpha=0.6, color='b')
plt.title("Histogram of Simulated Data vs. True Distribution")
plt.xlabel("x")
plt.ylabel("Density")
plt.show()
```

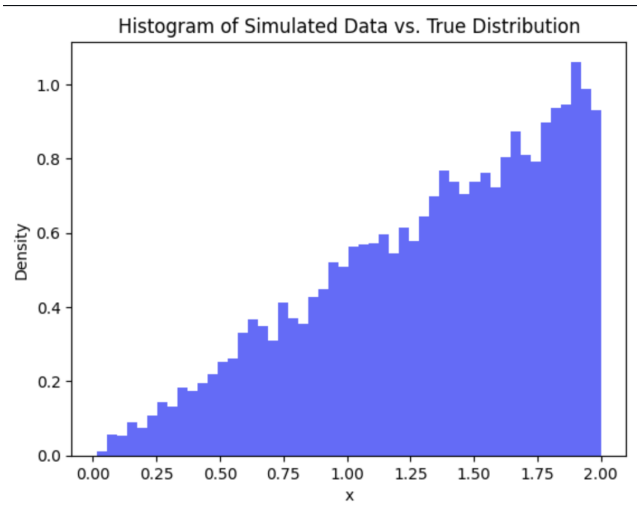

Figure 1: HISTOGRAM

Problem 2: Finite Mixture Approach

The given PDF is a mixture of two exponential distributions:

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

A sample is generated by selecting a component with the given probabilities and drawing from the corresponding exponential distribution. The Python implementation is:

```
def sample_mixture(n):
    U = np.random.uniform(0, 1, n)
    X = np.where(U < 2/3, np.random.exponential(1/2, n), np.random.exponential(1/3, n))
    return X

samples = sample_mixture(10000)
plt.hist(samples, bins=50, density=True, alpha=0.6, color='r')
plt.title("Histogram of Simulated Data")
plt.xlabel("x")
plt.ylabel("Density")
plt.show()
```

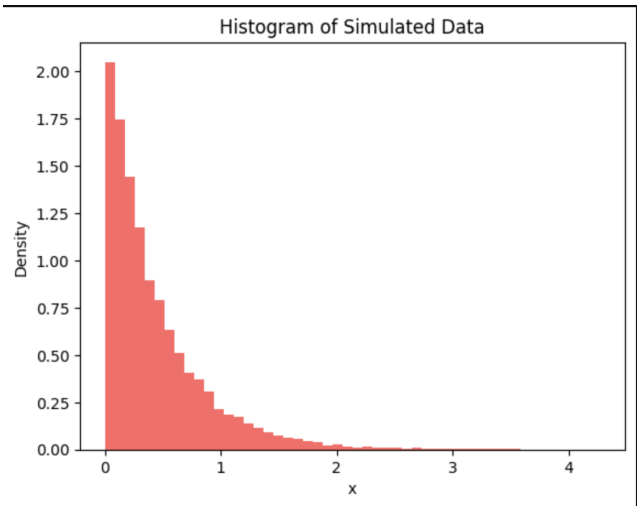

Figure 2: PLOT

Problem 3: Beta(3,3) Using Accept-Reject Algorithm

The target distribution is Beta(3,3), and we use an acceptance-rejection method with a proposal distribution. The implementation is as follows:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

def accept_reject_beta(n, alpha=3, beta_param=3):
    """
    Generate samples from Beta(alpha, beta) using the accept-reject method.

    Parameters:
    n (int): Number of samples to generate
    alpha (float): Alpha parameter of Beta distribution
    beta_param (float): Beta parameter of Beta distribution

    Returns:
    np.array: Array of samples from Beta(alpha, beta)
    """
    # Calculate the maximum value of Beta(3,3) PDF
    # For Beta(3,3), the max occurs at x = 0.5
    max_height = beta.pdf(0.5, alpha, beta_param)
```

```
    samples = []
    attempts = 0

    while len(samples) < n:
        # Generate uniform proposals
        x = np.random.uniform(0, 1)  # Proposed x value
        y = np.random.uniform(0, max_height)  # Proposed height

        # Accept or reject based on the PDF value
        if y <= beta.pdf(x, alpha, beta_param):
            samples.append(x)

        attempts += 1

    # Calculate acceptance rate
    acceptance_rate = n / attempts
    print(f"Accept-reject efficiency: {acceptance_rate:.4f}")

    return np.array(samples)

# Generate 500 samples from Beta(3,3)
samples = accept_reject_beta(500)

# Calculate sample statistics
sample_mean = np.mean(samples)
sample_var = np.var(samples)

# Calculate theoretical values
alpha = 3
beta_param = 3
true_mean = alpha / (alpha + beta_param)
true_var = (alpha * beta_param) / ((alpha + beta_param)**2 * (alpha + beta_param + 1))

# Print results
print(f"Sample Mean: {sample_mean:.6f}")
```

```
print(f"Sample Variance: {sample_var:.6f}")
print(f"True Mean: {true_mean:.6f}")
print(f"True Variance: {true_var:.6f}")

# Plot the histogram of samples
plt.figure(figsize=(10, 6))
plt.hist(samples, bins=30, density=True, alpha=0.7, label='Simulated Samples')

# Plot the true Beta(3,3) PDF
x = np.linspace(0, 1, 1000)
plt.plot(x, beta.pdf(x, alpha, beta_param), 'r-', lw=2, label='True Beta(3,3) PDF')

plt.title('Beta(3,3) Distribution: Accept-Reject Samples vs. True PDF')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

This script generates samples from the Beta(3,3) distribution, computes the mean and variance, and compares them to theoretical values.

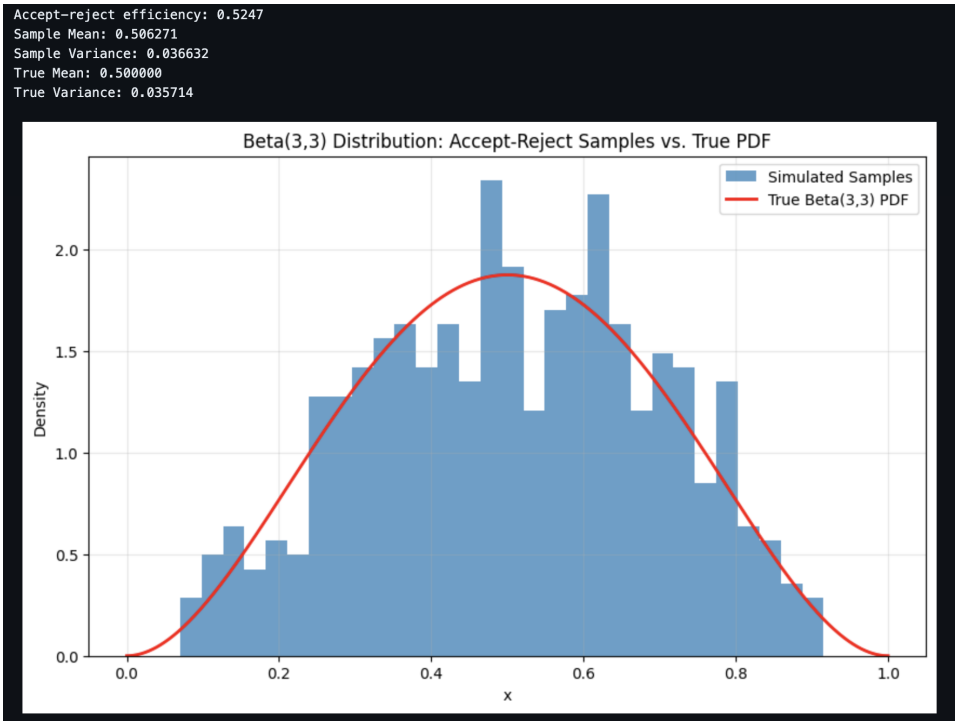

Figure 3: OUTPUT