# Exercise_round_7_2b

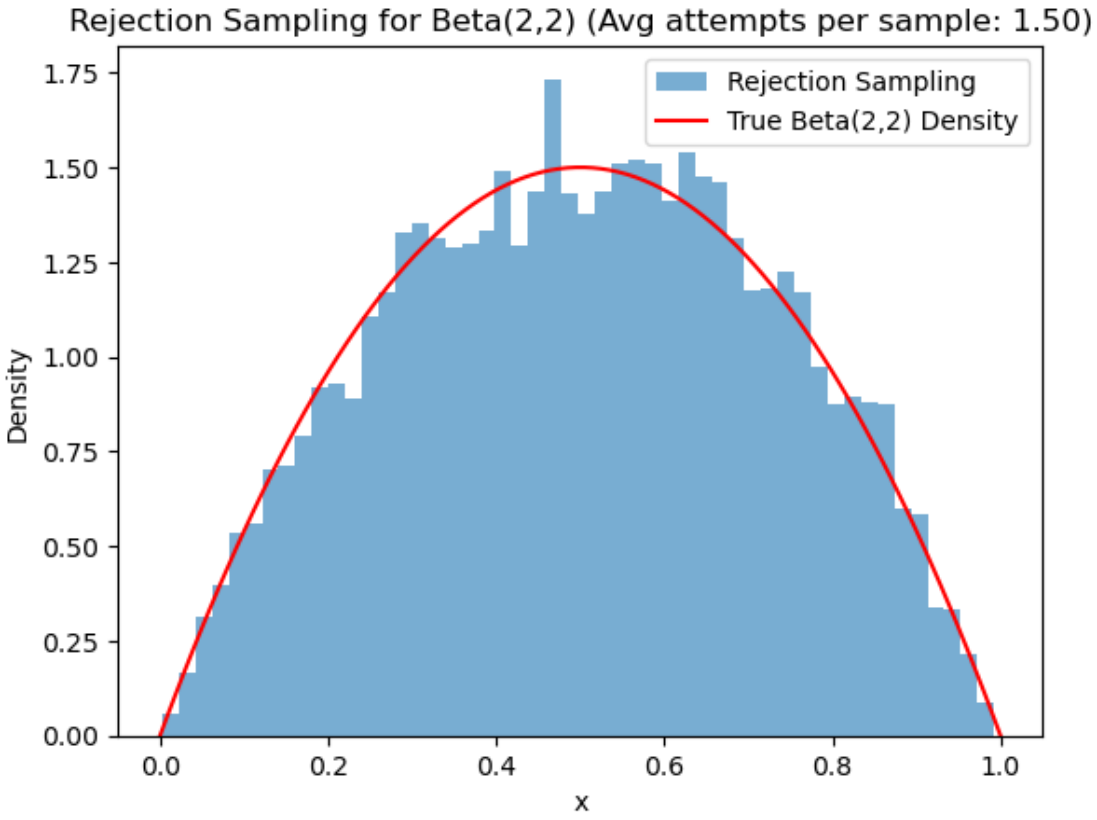
March 6, 2025

## 0.1 Exercise 2. (Rejection Sampling for Beta distribution)

(b) Implement the rejection sampling algorithm using the above proposal distribution and bound. Check numerically that the expected number of rounds for acceptance is M. Also check that the histogram of the samples matches the true density.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

# Define the Beta(2,2) density function
def beta_density(x):
    return 6 * x * (1 - x)  # Since q(x) = 6x(1-x) for Beta(2,2)

# Rejection sampling algorithm
def rejection_sampling(N):
    samples = []  # Store accepted samples
    M = 1.5  # Upper bound found earlier
    num_attempts = 0  # Count total attempts

    while len(samples) < N:
        x_star = np.random.uniform(0, 1)  # Sample from proposal  (x) = U(0,1)
        u = np.random.uniform(0, 1)  # Sample a uniform variable for rejection␣
 ↪test

        if u <= beta_density(x_star) / M:  # Accept condition
            samples.append(x_star)

        num_attempts += 1  # Count attempts

    return np.array(samples), num_attempts / N  # Return samples and avg␣
 ↪attempts per accept

# Generate samples
N_samples = 10000
samples, avg_attempts = rejection_sampling(N_samples)

# Plot histogram of samples
```

```
plt.hist(samples, bins=50, density=True, alpha=0.6, label="Rejection Sampling")
x_vals = np.linspace(0, 1, 100)
plt.plot(x_vals, beta.pdf(x_vals, 2, 2), 'r-', label="True Beta(2,2) Density")
plt.xlabel("x")
plt.ylabel("Density")
plt.legend()
plt.title(f"Rejection Sampling for Beta(2,2) (Avg attempts per sample:␣
  ↪{avg_attempts:.2f})")
plt.show()
```




[ ]: