Assignment 1

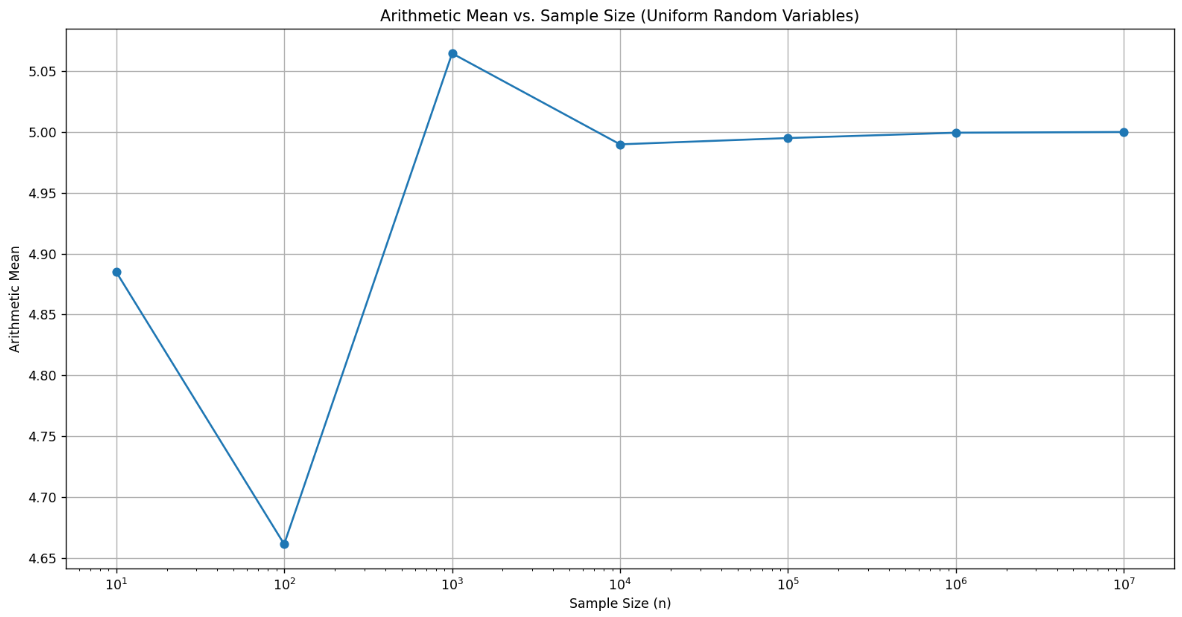
Q 1) Draw n uniform random variables (X_k) in the interval [0, 10]. For n = 10, 100, 1000, 10000, 100000, 1000000, 10000000 compute the arithmetic means of the random variable. Plot in the logarithmic scale (mean vs n).

Cause : Law of Large Numbers

Code

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

# Function to generate uniform random variables and compute their mean
def generate_uniform_means(n_values):
    means = []
    for n in n_values:
        random_variables = [random.uniform(0, 10) for _ in range(n)]
        mean = sum(random_variables) / n
        means.append(mean)
    return means

# Sample sizes
n_values = [10, 100, 1000, 10000, 100000, 1000000, 10000000]

# Generate means
means = generate_uniform_means(n_values)

# Plotting
plt.figure()
plt.plot(n_values, means, marker='o')
plt.xscale('log')  # Set x-axis to logarithmic scale
plt.xlabel('Sample Size (n)')
plt.ylabel('Arithmetic Mean')
plt.title('Arithmetic Mean vs. Sample Size (Uniform Random Variables)')
plt.grid(True)
plt.show()
```

Plots



Q 2) In the same experiment, shift and scale the random variable values such that the resulting random variables are of zero mean and unity variance. Then for n = 1000, compute (1/sqrt(n))* Sum (X_k). This random variable is say Y_j. Compute value of Y_j for 10000 repeated experiments. Plot the histogram (approximate density function) generated by bin filling.

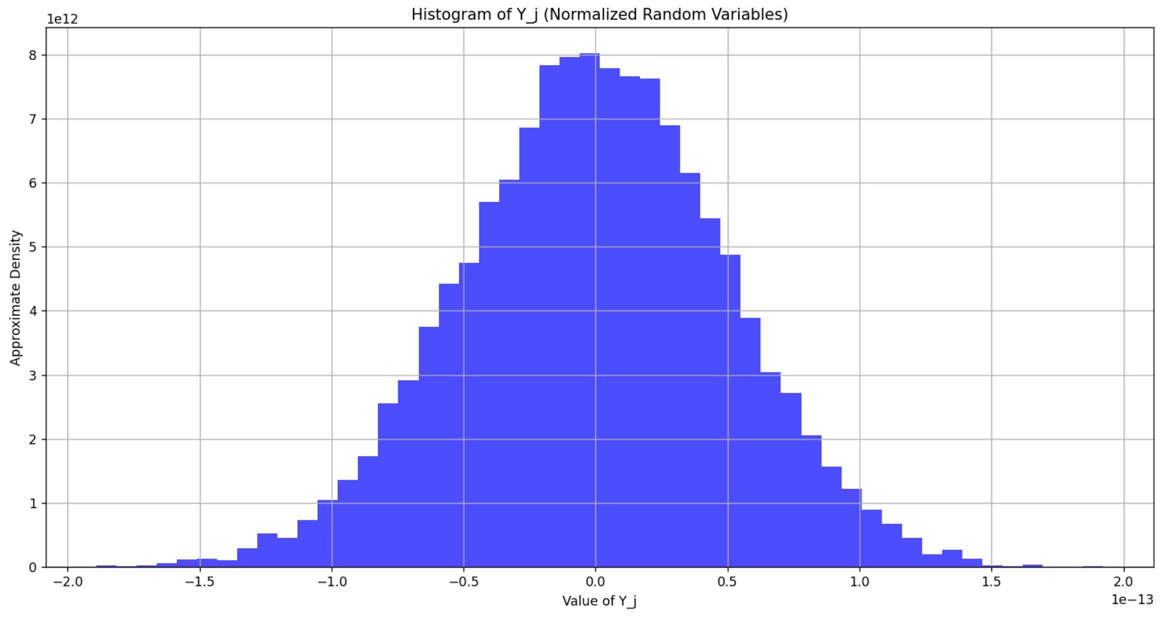
Cause : Central Limit Theorem

code

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

# Function to generate and normalize random variables
```

```
def generate_normalized_variables(n, num_experiments):
    normalized_variables = []
    for _ in range(num_experiments):
        random_variables = [random.uniform(0, 10) for _ in range(n)]
        mean = sum(random_variables) / n
        variance = sum((x - mean) ** 2 for x in random_variables) / (n - 1)
        normalized = [(x - mean) / np.sqrt(variance) for x in random_variables]
        y_j = (1 / np.sqrt(n)) * sum(normalized)
        normalized_variables.append(y_j)
    return normalized_variables

# Parameters
n = 1000
num_experiments = 10000

# Generate normalized variables and compute Y_j
normalized_variables = generate_normalized_variables(n, num_experiments)

# Plotting histogram
plt.figure()
plt.hist(normalized_variables, bins=50, density=True, color='blue', alpha=0.7)
plt.xlabel('Value of Y_j')
plt.ylabel('Approximate Density')
plt.title('Histogram of Y_j (Normalized Random Variables)')
plt.grid(True)
plt.show()
```

Plots



Q 3) Consider a box function B(x) in the interval

[-1,1]. Compute the repetitive convolution of B(x) with itself n times

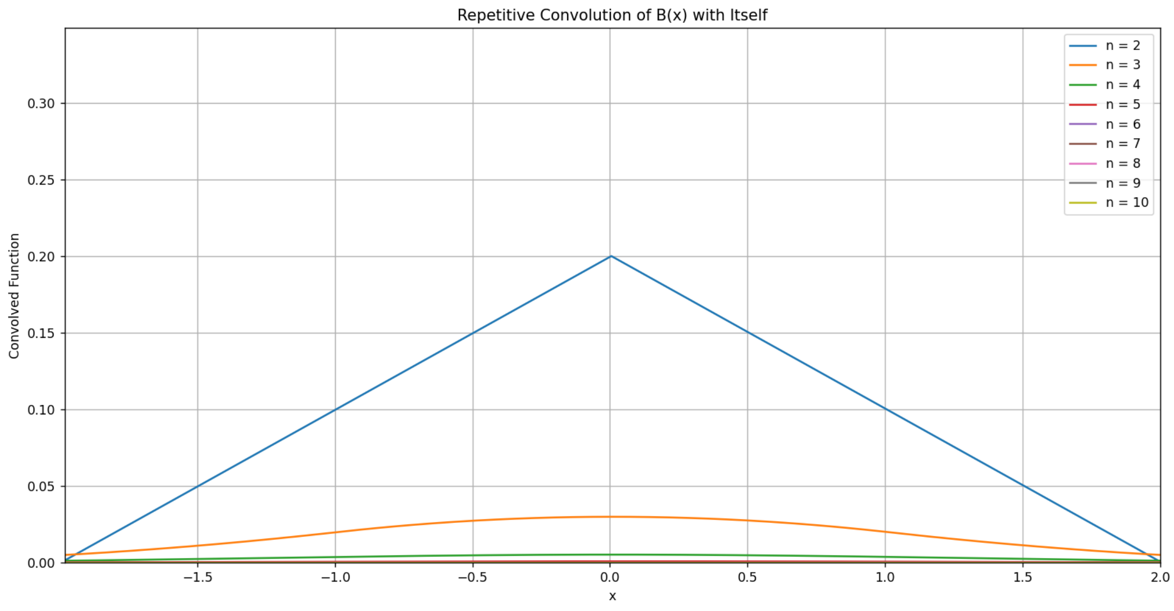
and plot it. Do it for n = 2, 3, 4, 5, 6, 7, 8, 9, 10.

Cause : Central Limit Theorem

code

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

# Define the box function B(x)
def box_function(x):
    return np.where(np.logical_and(x >= -1, x <= 1), 1, 0)

# Define the interval and x values for plotting
x = np.linspace(-5, 5, 1000)

# Convolve the box function with itself n times and plot
n_values = [2, 3, 4, 5, 6, 7, 8, 9, 10]

plt.figure(figsize=(12, 8))

for n in n_values:
    convolved_y = box_function(x)
    for _ in range(n - 1):
        convolved_y = np.convolve(convolved_y, box_function(x), mode='same') / len(x)
    plt.plot(x, convolved_y, label=f'n = {n}')

plt.xlabel('x')
plt.ylabel('Convolved Function')
plt.title('Repetitive Convolution of B(x) with Itself')
plt.legend()
plt.grid(True)
plt.xlim(-2, 2)
plt.ylim(0, 1.2)
plt.show()
```

Plots



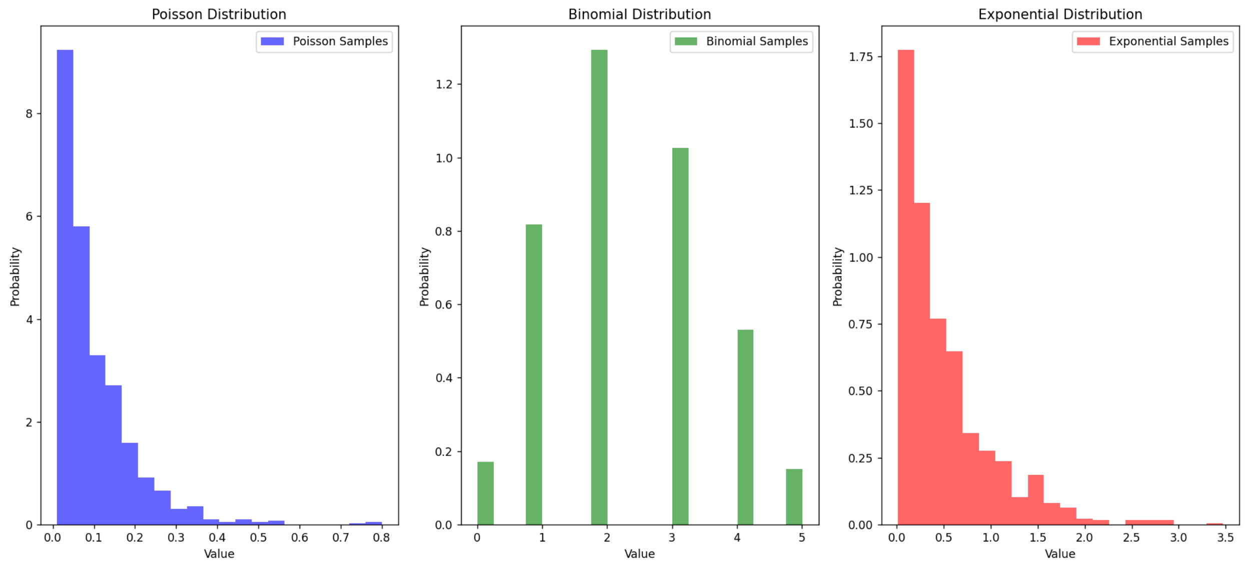
Q4

Draw uniform random variable in the interval [0,1]. Now using Monte Carlo inversion technique generate the following random variables and plot the histograms:

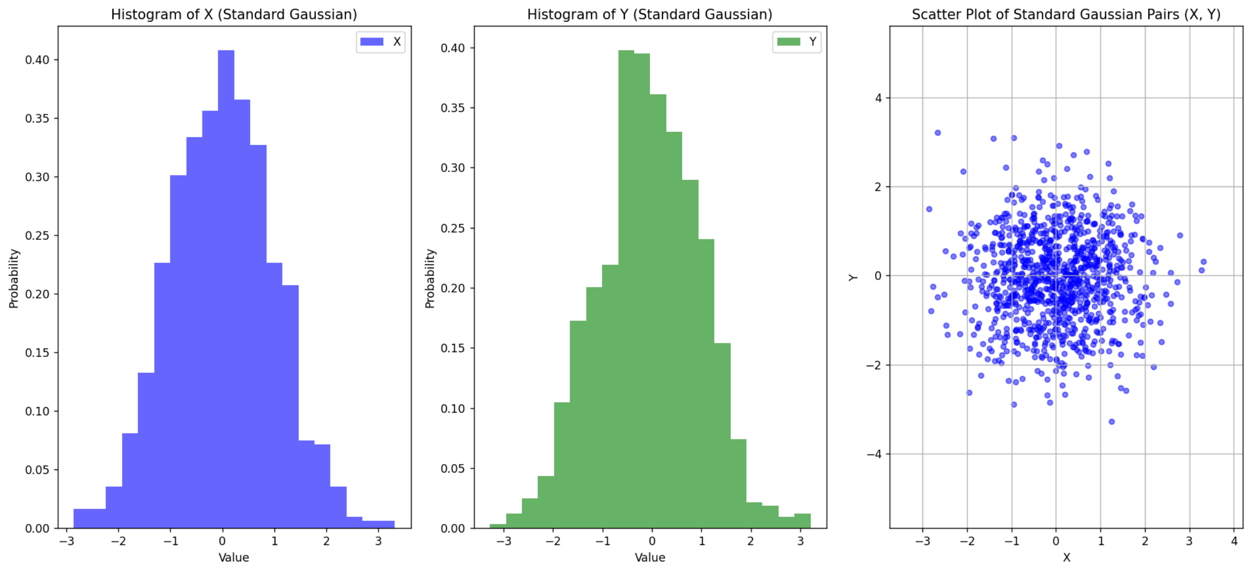
i) Binomial (p =0.4, n = 6)

ii) Poisson (lambda = 3)

iii) exponential (lambda = 2)



iv) Using Box Muller technique generate the pair of standard gaussian random variables X and Y.


Histogram of X (Standard Gaussian)
X
Probability
Value
Histogram of Y (Standard Gaussian)
Y
Probability
Value
Scatter Plot of Standard Gaussian Pairs (X, Y)
Y
X