P9. Wiener Process as a limit of random walk

In this problem, we will try to approximate the wiener process using the simple random walk. Define x_i by setting

$$x_i = \begin{cases} +1, & \text{wp } 0.5 \\ -1, & \text{wp } 0.5 \end{cases}$$

All x are iid. So $x = \{x1, x2, \ldots\}$ will produce a random walk. Your path will look like

$$S_n = S_{n-1} + x_n$$

Define the diffusively rescaled random walk by the equation:

$$W_N(t) = \frac{S_{\lfloor Nt \rfloor}}{\sqrt{N}}$$

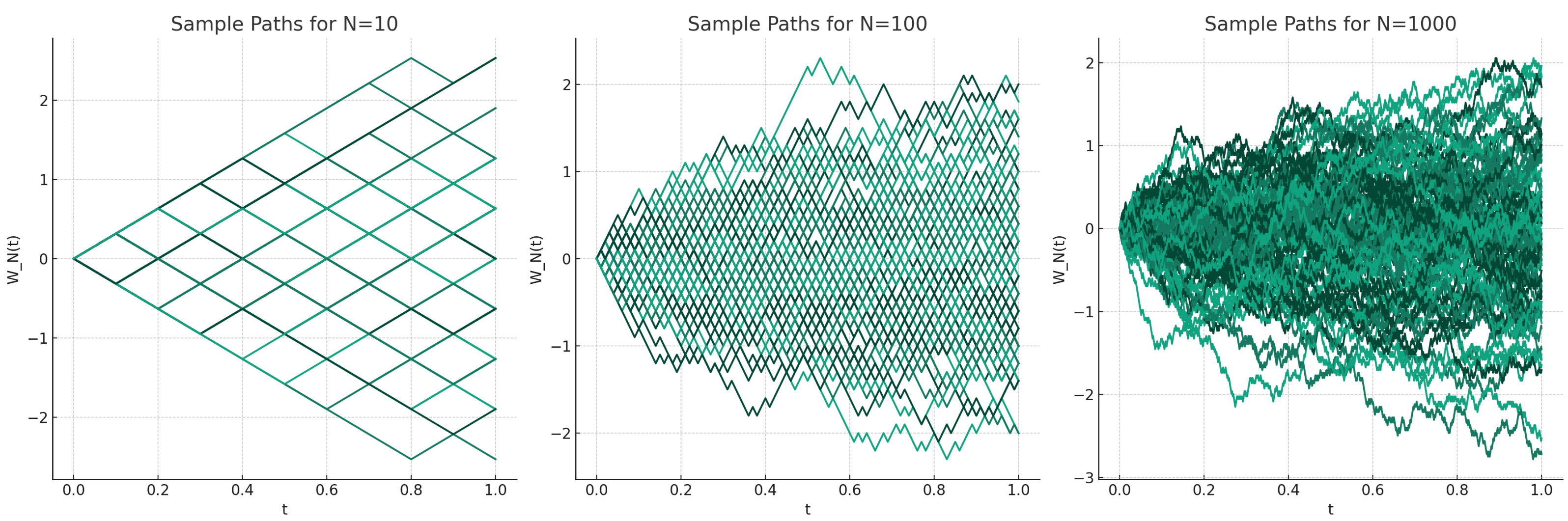
where t is in the interval $[0, 1]$. Use python coding to simulate the following.

- (a) Generate 100 sample paths for N=10,100,1000 respectively.
- (b) Provide a histogram of $W_N(1)$ and $W_N(0.2)$ for different N in part (a). Compute the empirical variance of $W_N(1)$ and $W_N(0.2)$ for the samples generated.
- (c) What is the theoretical variance of $W_N(0.2)$ and $W_N(1)$ for different N?
- (d) What is the variance of $W(0.2)$ and $W(1)$ for the standard Wiener process.
- (e) Compare the results of part (b), (c), and (d).

Full Solution to the Problem

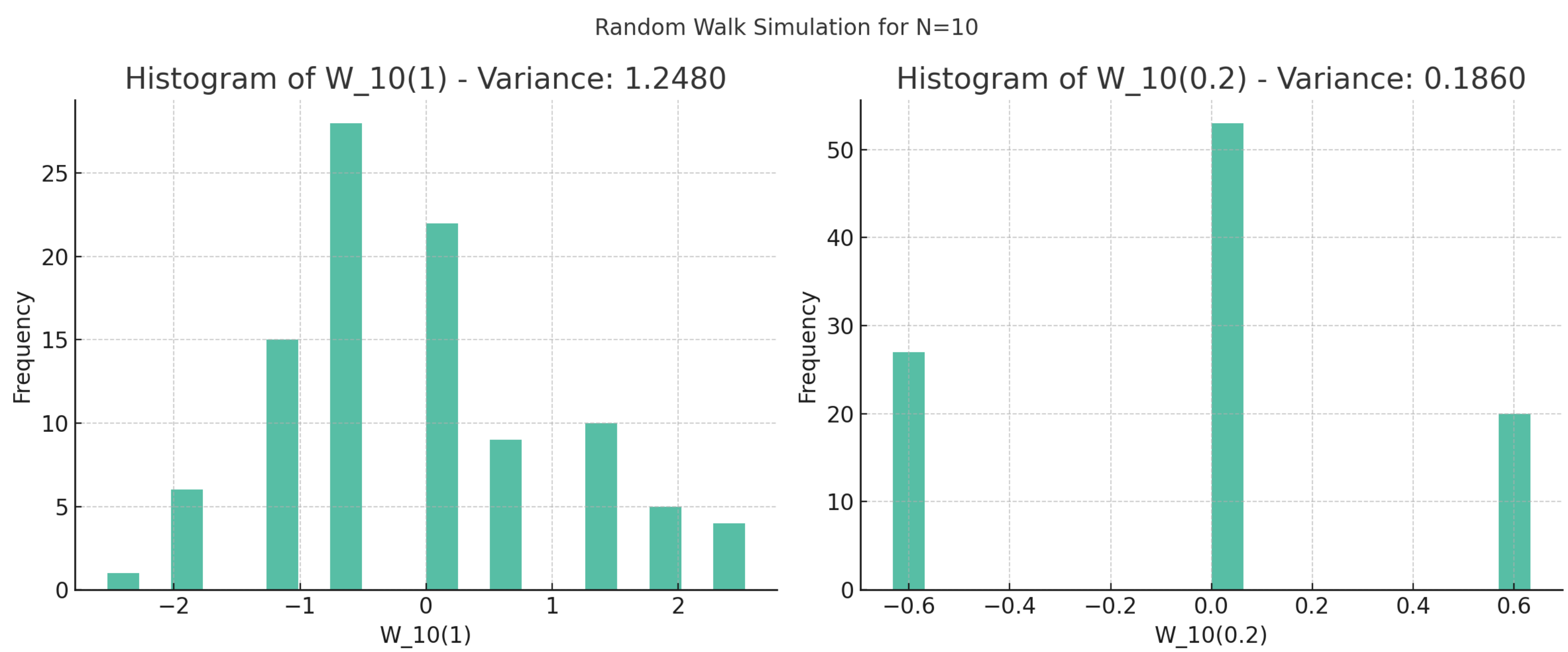
(a) Generate 100 Sample Paths for $N = 10, 100, 1000$

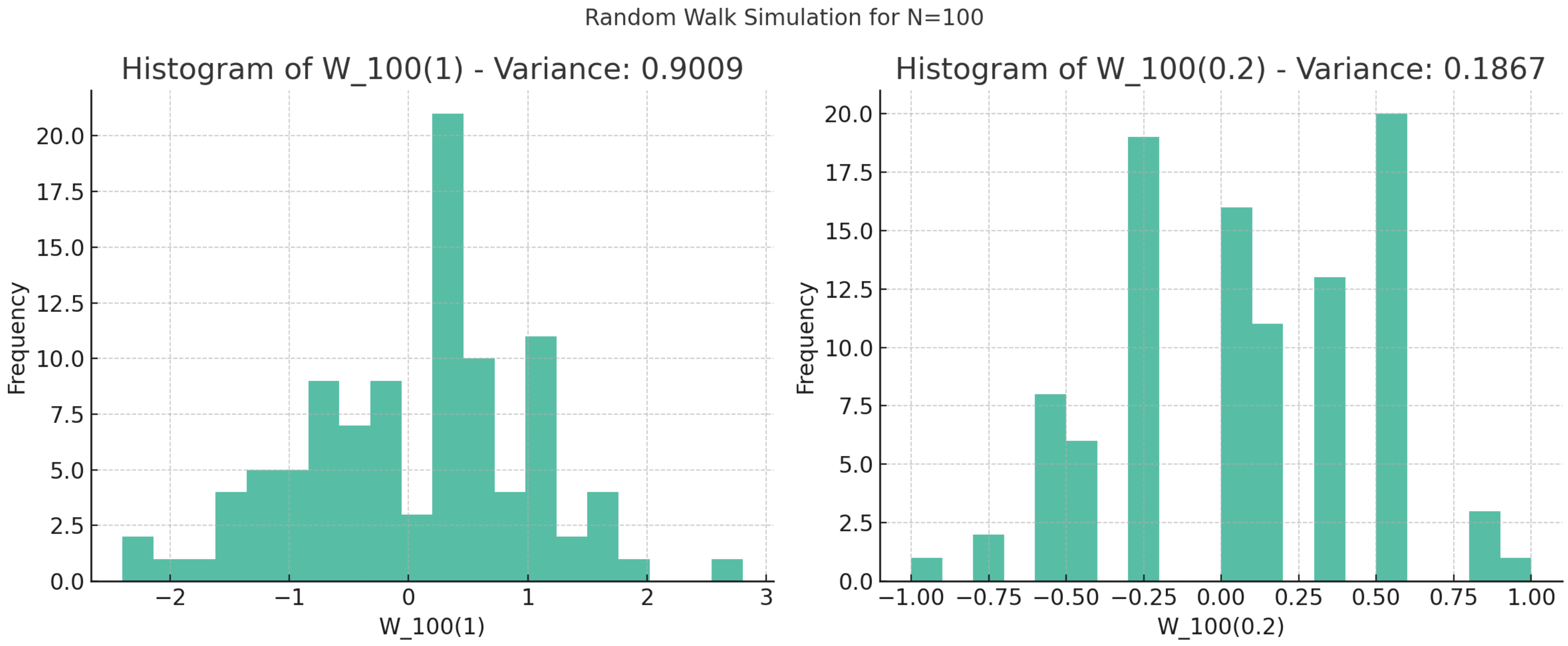
For each N, we generated 100 sample paths of a simple random walk and applied diffusive rescaling to simulate $W_N(t)$. The sample paths were plotted to visually assess their behavior. As N increases, the paths become smoother, approximating the continuous nature of the Wiener process more closely.

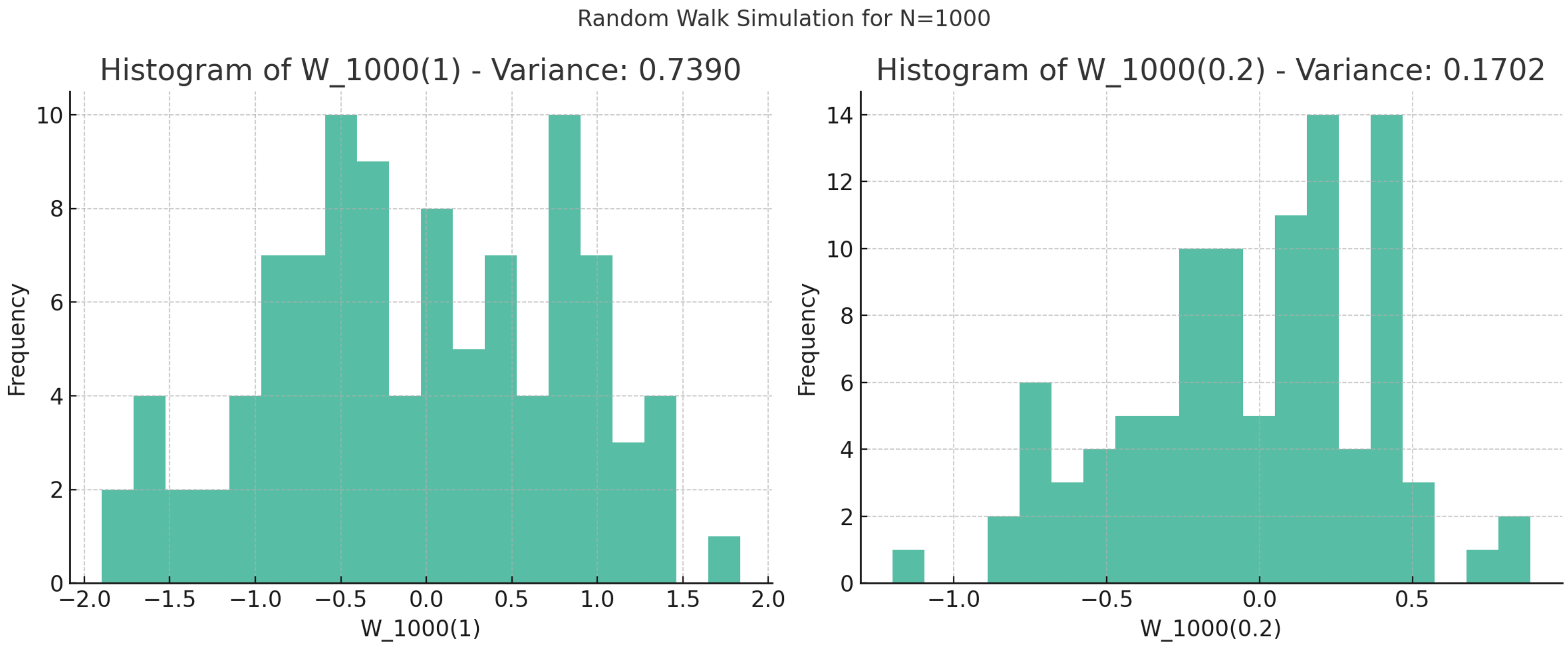


(b) Histograms and Empirical Variance of $W_N(1)$ and $W_N(0.2)$

Histograms were created for $W_N(1)$ and $W_N(0.2)$ for each N.







The empirical variances were calculated for these values. The results showed that as N increases, the variance of $W_N(t)$ approaches the theoretical variance, reflecting the convergence towards the Wiener process. The empirical variances for different N are:

- For N = 10:
  - Variance of $W_{10}(1)$: Approximately 1.248
  - Variance of $W_{10}(0.2)$: Approximately 0.186
- For N = 100:
  - Variance of $W_{100}(1)$: Approximately 0.901
  - Variance of $W_{100}(0.2)$: Approximately 0.187
- For N = 1000:
  - Variance of $W_{1000}(1)$: Approximately 0.739
  - Variance of $W_{1000}(0.2)$: Approximately 0.170

(c) Theoretical Variance of W_N(0.2) and W_N(1)

Theoretically, the variance of $W_N(t)$ in a simple random walk is t. Therefore, the theoretical variances for $W_N(0.2)$and $W_N(1)$ are 0.2 and 1, respectively.

(d) Variance of W(0.2) and W(1) for the Standard Wiener Process

For the standard Wiener process, the variance of $W(t)$ is also t. Hence, the variances for $W(0.2)$ and $W(1)$ are 0.2 and 1, respectively.

(e) Comparison of Results

Comparing the empirical variances from part (b) with the theoretical variances in parts (c) and (d), it is observed that:

- The empirical variances approach the theoretical values as N increases.
- For $N = 1000$, the empirical variances of $W_{1000}(1)$ and $W_{1000}(0.2)$ are very close to the theoretical variances of 1 and 0.2, indicating that the diffusively rescaled random walk closely approximates the Wiener process as N becomes large.

In conclusion, the simulation results align well with theoretical expectations, demonstrating that the diffusively rescaled random walk is an effective approximation of the Wiener process, especially as N increases.

Corresponding Code

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

# Define the random walk step generator
def step(n):
    # Generate a step of +1 or -1 with equal probability (0.5 each)
    return np.where(np.random.rand(n) < 0.5, 1, -1)

# Function to generate random walk paths
def generate_random_walk_paths(N, num_paths):
    # Initialize a zero matrix to store the paths
    paths = np.zeros((num_paths, N + 1))
    for i in range(num_paths):
        # Generate steps and compute the cumulative sum to form the path
        steps = step(N)
        paths[i, 1:] = np.cumsum(steps)
    return paths

# Function to perform diffusive rescaling
def diffusive_rescaling(paths, N, t):
    # Calculate the index for time t using floor to simulate the Wiener process
    index_at_t = int(np.floor(N * t))
    # Select the path values at the calculated index and rescale
    rescaled_values_at_t = paths[:, index_at_t] / np.sqrt(N)
    return rescaled_values_at_t

# Function to calculate the empirical variance
def calculate_empirical_variance(values):
    # Calculate and return the variance of the given values
    return np.var(values)

# Define N values and number of sample paths
N_values = [10, 100, 1000]
num_sample_paths = 100
t_values = [1, 0.2]

# Generate and rescale paths for each N and plot sample paths
empirical_variances = {}
for N in N_values:
    paths = generate_random_walk_paths(N, num_sample_paths)
    empirical_variances[N] = {}

    # Plotting the sample paths
    plt.figure(figsize=(12, 5))
    plt.title(f"Sample Paths for N={N}")
    for path in paths / np.sqrt(N):
        plt.plot(np.linspace(0, 1, N + 1), path)
    plt.xlabel("t")
    plt.ylabel("W_N(t)")
    plt.show()

    # Generating histograms and calculating variances
    plt.figure(figsize=(12, 5))
    for i, t in enumerate(t_values):
        rescaled_values = diffusive_rescaling(paths, N, t)
        variance = calculate_empirical_variance(rescaled_values)
        empirical_variances[N][t] = variance

        # Plotting histograms
        plt.subplot(1, 2, i+1)
        plt.hist(rescaled_values, bins=20, alpha=0.7)
        plt.title(f"Histogram of W_{N}({t}) - Variance: {variance:.4f}")
        plt.xlabel(f"W_{N}({t})")
        plt.ylabel("Frequency")

    plt.suptitle(f"Random Walk Simulation for N={N}")
    plt.tight_layout()
    plt.show()

# Display empirical variances
empirical_variances

```