Stat 342 HW/LAB3

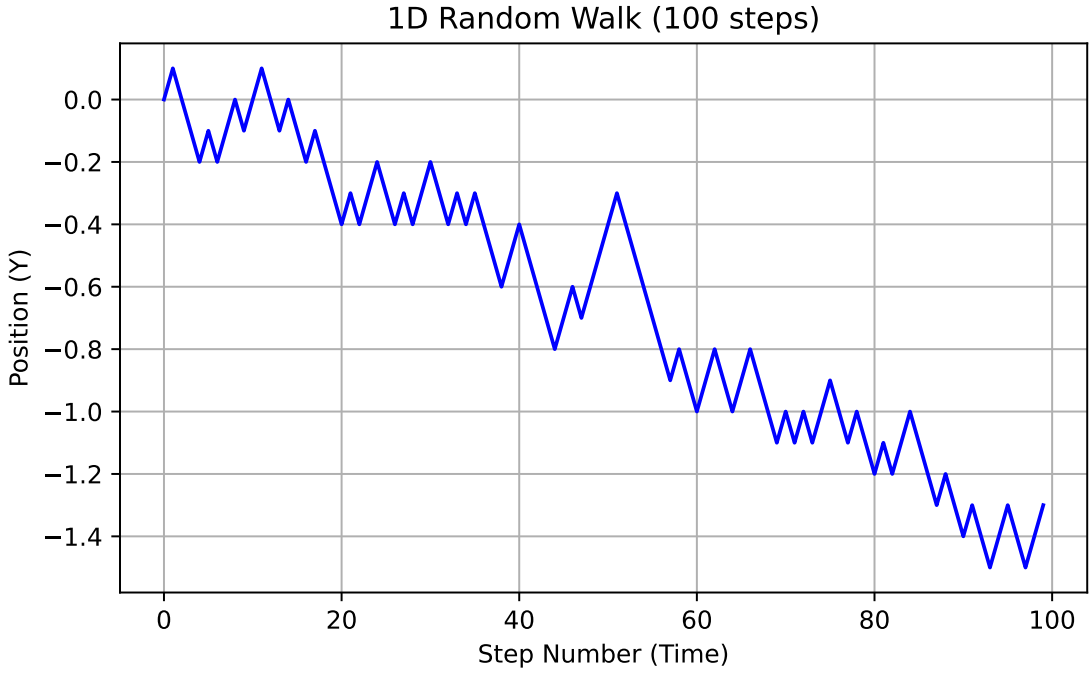
Blayne Downer & Ben Liu

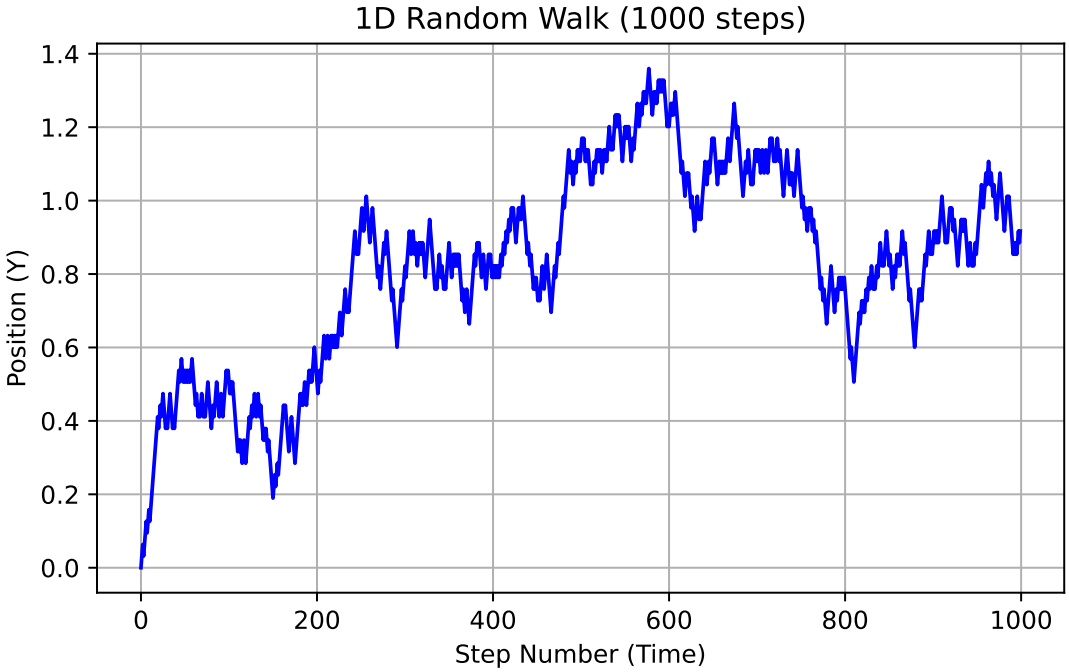
2025-05-09

Question 1

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q1(n):

    stepsize = 1/np.sqrt(n)
    x= np.zeros(n)
    y= np.zeros(n)

    for i in range(n):
        x[i] = i
        if i == 0:
            y[i] = 0
        else:
            y[i] += np.random.choice([-1,1])*stepsize + y[i-1]

    plt.plot(x, y, 'b-')
    plt.title(f"1D Random Walk ({n} steps)")
    plt.xlabel("Step Number (Time)")
    plt.ylabel("Position (Y)")
    plt.grid(True)
    plt.show()
    #plt.clf()



if __name__ == "__main__":

    random_walk_q1(10)
    random_walk_q1(100)
    random_walk_q1(1000)
    random_walk_q1(10000)
```


1D Random Walk (10 steps)
Position (Y)
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
2
4
6
8
Step Number (Time)

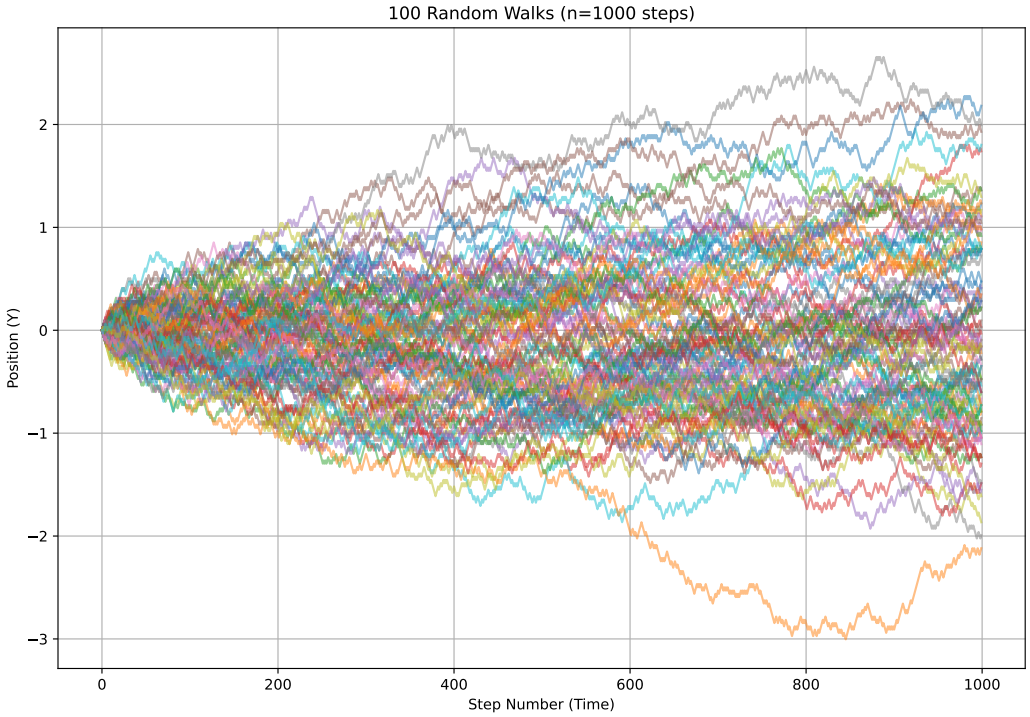

1D Random Walk (100 steps)
Position (Y)
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
0
20
40
60
80
100
Step Number (Time)



Question 2

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)
```

```python
def random_walk_q2(num_simulations):
    n = 1000
    stepsize = 1 / np.sqrt(n)

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

    for _ in range(num_simulations):
        x = np.zeros(n)
        y = np.zeros(n)

        for i in range(n):
            x[i] = i
            if i > 0:
                y[i] = y[i-1] + np.random.choice([-1, 1]) * stepsize

        plt.plot(x, y, alpha=0.5)

    plt.title(f"{num_simulations} Random Walks (n={n} steps)")
    plt.xlabel("Step Number (Time)")
    plt.ylabel("Position (Y)")
    plt.grid(True)
    plt.show()



if __name__ == "__main__":

    random_walk_q2(100)
```

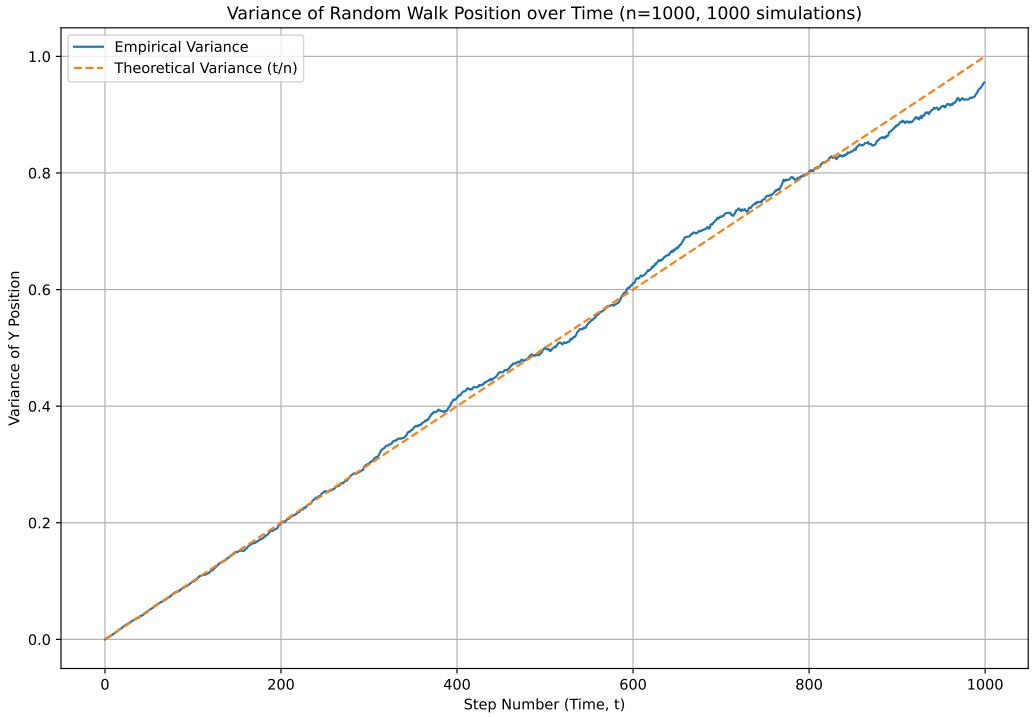


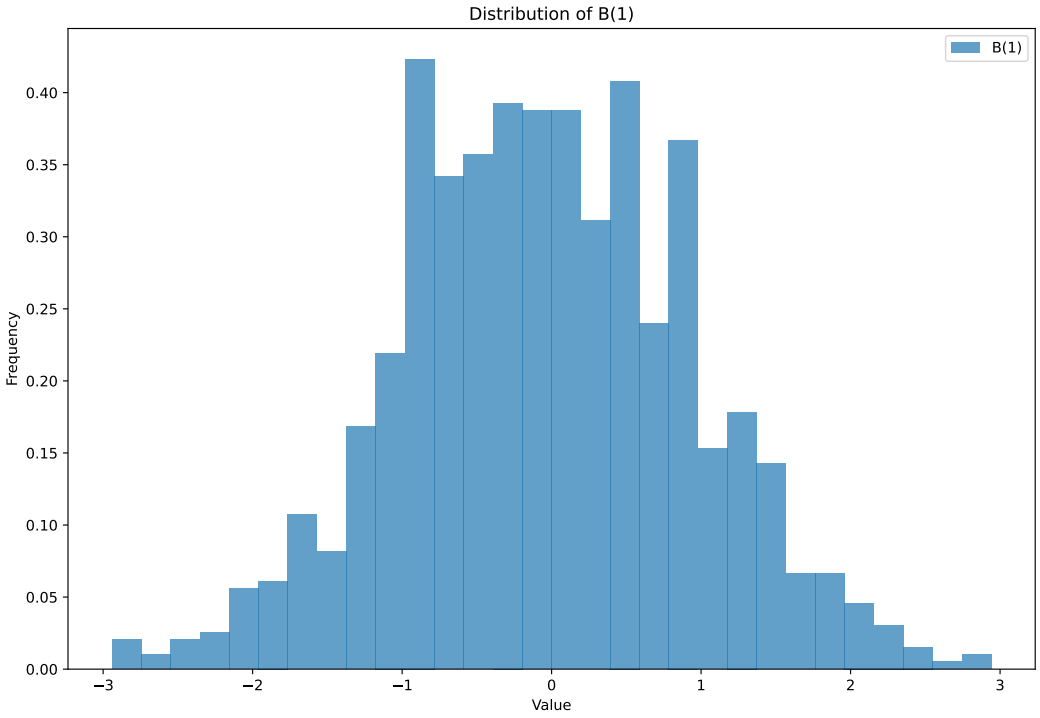
Question 3-5

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q3(num_simulations):

    n = 1000
    stepsize = 1 / np.sqrt(n)

    all_y_positions = np.zeros((num_simulations, n))

    x_steps = np.arange(n)

    for sim_index in range(num_simulations):
        y = np.zeros(n)

        for i in range(1, n):
            y[i] = y[i-1] + np.random.choice([-1, 1]) * stepsize

        all_y_positions[sim_index, :] = y

    empirical_variances = np.var(all_y_positions, axis=0)
```

```
    theoretical_variances = x_steps * (stepsize**2)

    plt.figure(figsize=(12, 8))
    plt.plot(x_steps, empirical_variances, label='Empirical Variance')
    plt.plot(x_steps, theoretical_variances, label='Theoretical Variance (t/n)', linestyle='--')

    plt.title(f"Variance of Random Walk Position over Time (n={n}, {num_simulations} simulations)")
    plt.xlabel("Step Number (Time, t)")
    plt.ylabel("Variance of Y Position")
    plt.legend()
    plt.grid(True)
    plt.show()

    idx1, idx2 = 300, 600
    if n > idx1 and n > idx2:
        y_at_idx1 = all_y_positions[:, idx1]
        y_at_idx2 = all_y_positions[:, idx2]
        cov_matrix = np.cov(y_at_idx1, y_at_idx2, ddof=0)
        empirical_covariance_np = cov_matrix[0, 1]

        print(f"Empirical Covariance (s=0.3, t=0.6): {empirical_covariance_np:.6f}")

    plt.figure(figsize=(12, 8))
    plt.hist(all_y_positions[:, -1], bins=30, density=True, alpha=0.7, label='B(1)')
    plt.title('Distribution of B(1)')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.legend()
    plt.show()

if __name__ == "__main__":

    random_walk_q3(1000)
```

```
## Empirical Covariance (s=0.3, t=0.6): 0.297906
```





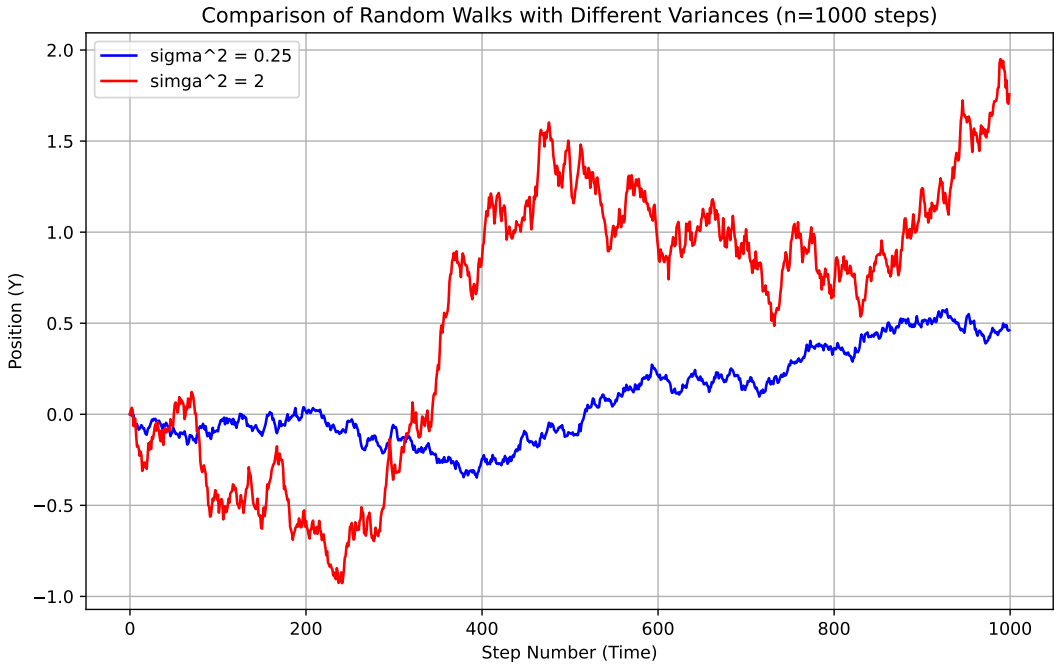
Q3) The lines appear overall pretty close, but towards the end, the empiral variance seems to waver away

from the theoretical variance.

Q5) I notice that the distribution appears to be (roughly) normal, which is expected.

Question 6

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q6(num_simulations, variance1, variance2=None):
    x = np.zeros(num_simulations)
    y1 = np.zeros(num_simulations)
    y2 = np.zeros(num_simulations)

    for i in range(num_simulations):
        x[i] = i
        if i == 0:
            y1[i] = 0
        else:
            y1[i] = y1[i-1] + np.random.normal(0, np.sqrt(variance1/num_simulations))

    if variance2 is not None:
        for i in range(num_simulations):
            if i == 0:
                y2[i] = 0
            else:
                y2[i] = y2[i-1] + np.random.normal(0, np.sqrt(variance2/num_simulations))

    plt.figure(figsize=(10, 6))
    plt.plot(x, y1, 'b-', label=f'sigma^2 = {variance1}')
    if variance2 is not None:
        plt.plot(x, y2, 'r-', label=f'simga^2 = {variance2}')

    plt.title(f"Comparison of Random Walks with Different Variances (n={num_simulations} steps)")
    plt.xlabel("Step Number (Time)")
    plt.ylabel("Position (Y)")
    plt.grid(True)
    plt.legend()
    plt.show()



if __name__ == "__main__":

    random_walk_q6(1000, 0.25, 2)
```

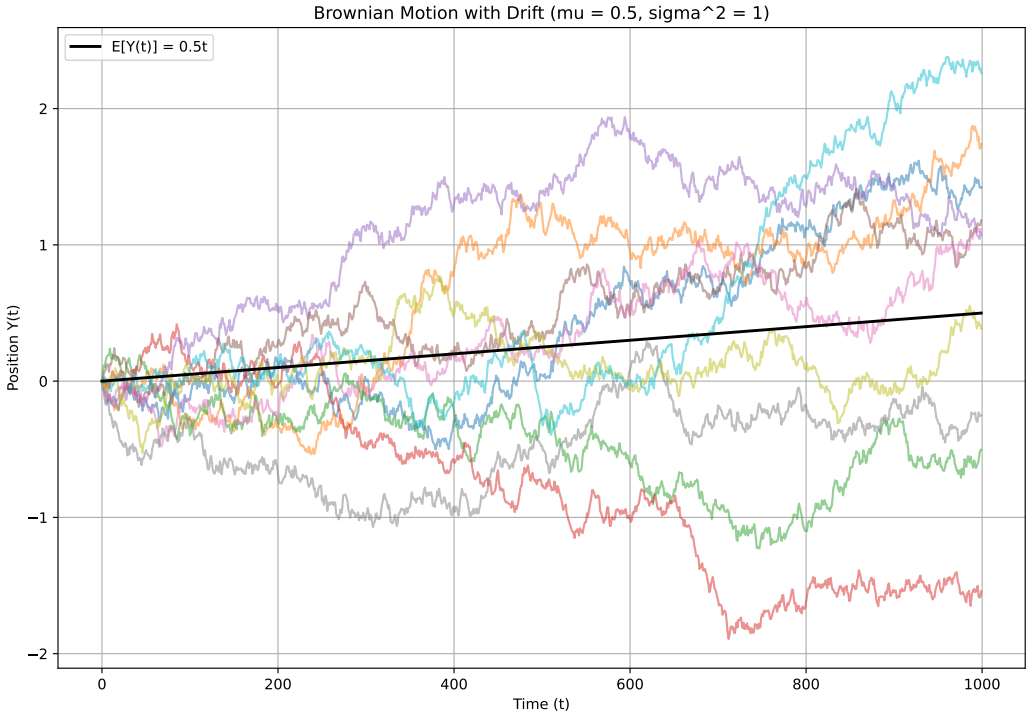


Compared to the previous CLT simulations, this one appears more smooth. I think we get the "smoothness" that we would see in a very large step number CLT simulation without all the steps.

Question 7

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q7(n, mean):
    plt.figure(figsize=(12, 8))

    for sim in range(10):
        x = np.zeros(n)
        y = np.zeros(n)

        for i in range(n):
            x[i] = i
            if i == 0:
                y[i] = 0
            else:
                y[i] = y[i-1] + np.random.normal(0, 1/np.sqrt(n)) + (mean/n) # SBM + drift term

        plt.plot(x, y, alpha=0.5)

    expected_line = mean * x/n
    plt.plot(x, expected_line, 'k-', linewidth=2, label=f'E[Y(t)] = {mean}t')
```

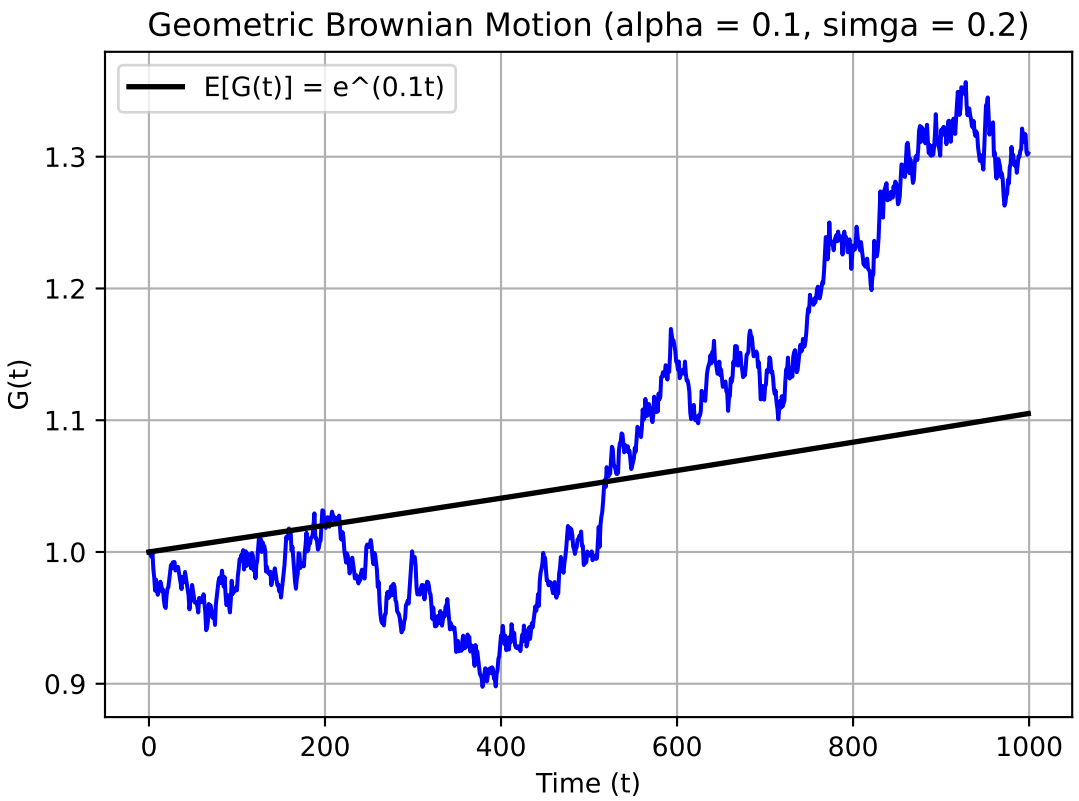
```
    plt.title(f"Brownian Motion with Drift (mu = {mean}, sigma^2 = 1)")
    plt.xlabel("Time (t)")
    plt.ylabel("Position Y(t)")
    plt.grid(True)
    plt.legend()
    plt.show()

if __name__ == "__main__":

    random_walk_q7(1000, 0.5)
```



Question 8

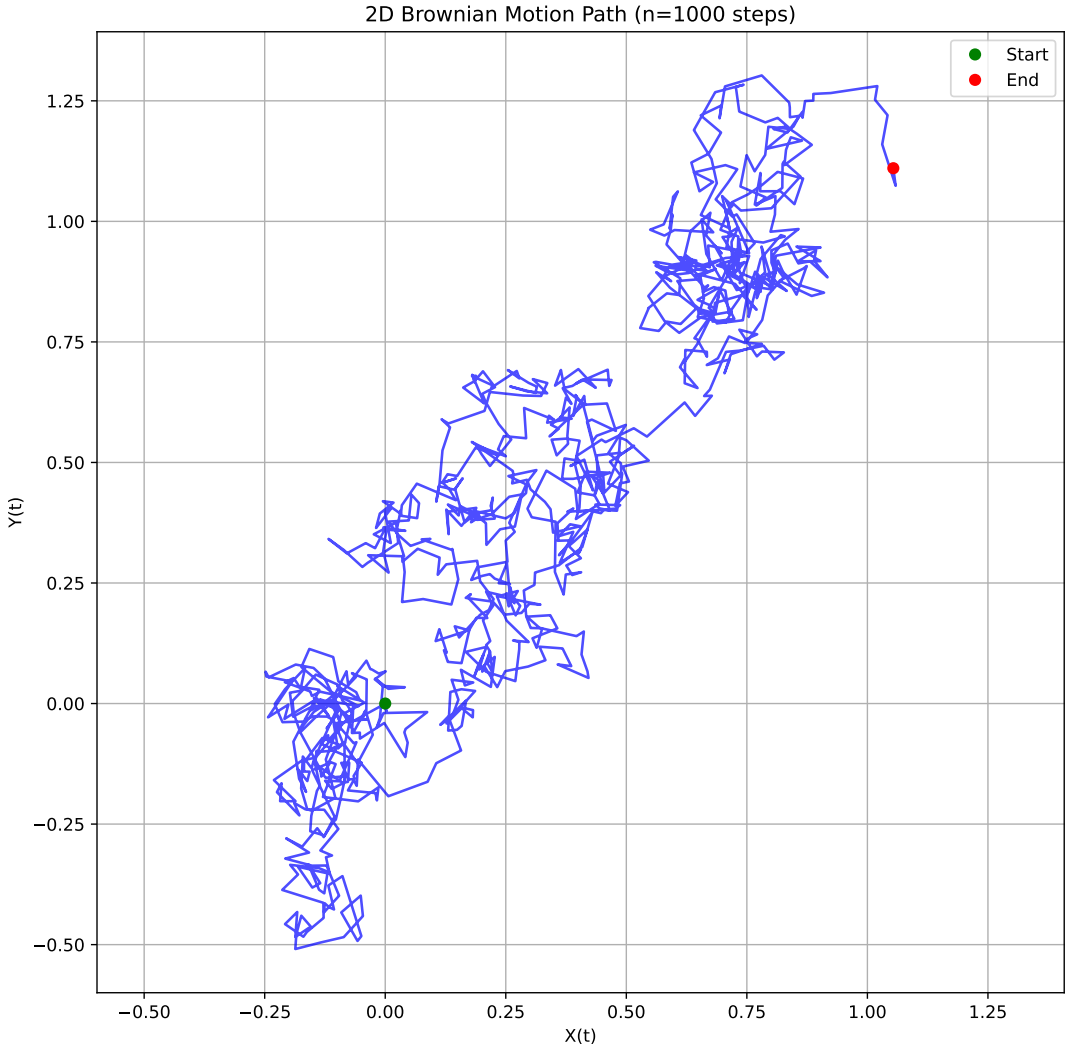
```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q8(n, alpha, sigma):
    x = np.zeros(n)
    y = np.zeros(n)
    B = np.zeros(n)

    for i in range(n):
```

```
        x[i] = i
        if i == 0:
            y[i] = 1  # G(0) = 1
            B[i] = 0
        else:
            B[i] = B[i-1] + np.random.normal(0, 1/np.sqrt(n))
            y[i] = np.exp((alpha - 0.5 * sigma**2) * (i/n) + sigma * B[i])

    plt.plot(x, y, 'b-')
    expected_line = np.exp(alpha * x/n)
    plt.plot(x, expected_line, 'k-', linewidth=2, label=f'E[G(t)] = e^({alpha}t)')

    plt.title(f"Geometric Brownian Motion (alpha = {alpha}, simga = {sigma})")
    plt.xlabel("Time (t)")
    plt.ylabel("G(t)")
    plt.grid(True)
    plt.legend()
    plt.show()


if __name__ == "__main__":

    random_walk_q8(1000, 0.1, 0.2)
```



Question 9

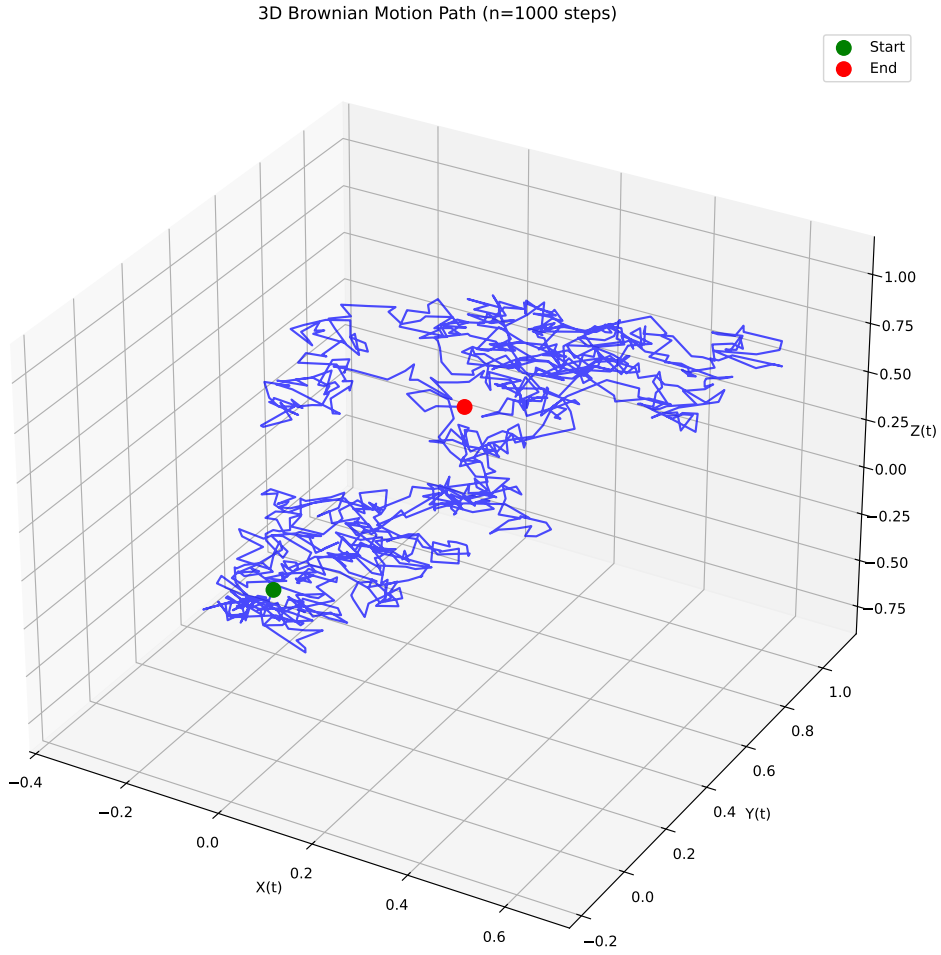
```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q9(n):
    x = np.zeros(n)
    y = np.zeros(n)

    for i in range(1, n):
        x[i] = x[i-1] + np.random.normal(0, 1/np.sqrt(n))
        y[i] = y[i-1] + np.random.normal(0, 1/np.sqrt(n))

    plt.figure(figsize=(10, 10))
    plt.plot(x, y, 'b-', alpha=0.7)
    plt.plot(x[0], y[0], 'go', label='Start')
    plt.plot(x[-1], y[-1], 'ro', label='End')

    plt.title(f"2D Brownian Motion Path (n={n} steps)")
    plt.xlabel("X(t)")
    plt.ylabel("Y(t)")
    plt.grid(True)
    plt.legend()
    plt.axis('equal')
    plt.show()


if __name__ == "__main__":

    random_walk_q9(1000)
```



Question 10

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(342)

def random_walk_q10(n):
    x = np.zeros(n)
    y = np.zeros(n)
    z = np.zeros(n)
```

```
    for i in range(1, n):
        x[i] = x[i-1] + np.random.normal(0, 1/np.sqrt(n))
        y[i] = y[i-1] + np.random.normal(0, 1/np.sqrt(n))
        z[i] = z[i-1] + np.random.normal(0, 1/np.sqrt(n))

    fig = plt.figure(figsize=(12, 12))
    ax = fig.add_subplot(111, projection='3d')

    ax.plot(x, y, z, 'b-', alpha=0.7)
    ax.scatter(x[0], y[0], z[0], color='green', s=100, label='Start')
    ax.scatter(x[-1], y[-1], z[-1], color='red', s=100, label='End')

    ax.set_title(f"3D Brownian Motion Path (n={n} steps)")
    ax.set_xlabel("X(t)")
    ax.set_ylabel("Y(t)")
    ax.set_zlabel("Z(t)")
    ax.legend()

    plt.show()


if __name__ == "__main__":

    random_walk_q10(1000)
```


3D Brownian Motion Path (n=1000 steps)
Start
End
X(t)
Y(t)
Z(t)
−0.4
−0.2
0.0
0.2
0.4
0.6
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75