Wiener Random Process Simulation

2016160145 박지혁

1. 이론설명

Wiener Process는 분자의 Brownian motion을 모델링한 Random Walk Stochastic Process의 일종이다. Wiener Process 는 다음과 같이 기술될 수 있다.

1.1. Discrete Wiener Process

h 의 time unit으로 M 의 magnitude step size를 갖는 $p = 1/2$인 1D random walk를 생각하자.

분자가 $X = 0$에서 출발할 때, 시간 t까지의 random step의 합 $X_h(t)$는 다음과 같이 정의한다.

$$X_h(t) = M(D[1] + D[2] + \cdots + D[n]) = MS[n], \qquad where\ n = \lfloor \frac{t}{h} \rfloor$$

이때, 각 time step에서의 discrete RV인 $D[k] = \pm 1$ 와 $D[k]$의 sum인 RV $S[n] \in [-n, n]$을 가정한다.

정의된 RV $X_h(t)$의 평균과 분산은 다음과 같다.

$$E[X_h(t)] = 0$$
$$Var[X_h(t)] = 0$$

또한 정의에 따라 $X_h(t)$의 점화식은 다음과 같다.

$$X_h(0) = 0$$
$$X_h(t+1) = X_h(t) + MD[n]$$

1.2. Continuous Wiener Process

위의 이산적인 상황으로부터 연속적인 상황으로의 Wiener Process를 유도하기 위해 time unit h와 magnitude step size M을 0으로 보내보자.

이때, 아주 짧은 시간 h에 순간 속도 v가 무한대가 되도록 하자.

즉, 다음을 가정한다.

$$M = \sqrt{\alpha h}$$
$$v = \frac{M}{h} = \sqrt{\frac{\alpha}{h}}$$
$$n \approx \frac{t}{h}$$

그러면 다음의 연속적인 RV $X(t)$를 생각할 수 있다.

$$X(t) \coloneqq \lim_{h\to 0} X_h(t) = \lim_{h\to 0} MS[n] = \lim_{n\to 0} \sqrt{\frac{\alpha t}{n}} S[n]$$

이때 $X(t)$의 평균과 분산은 다음과 같다.

$$E[X(t)] = 0$$
$$Var[X(t)] = M^2 \frac{t}{n} = \sqrt{\alpha h}^2 \frac{t}{h} = \alpha t$$

2. 코드

다음은 시뮬레이션을 위해 작성한 코드이다. 코드의 전문은 개인 github

(https://github.com/jihyuk1023/AI_physics/blob/main/WienerRandomProcess.ipynb)

에서도 확인 가능하다.

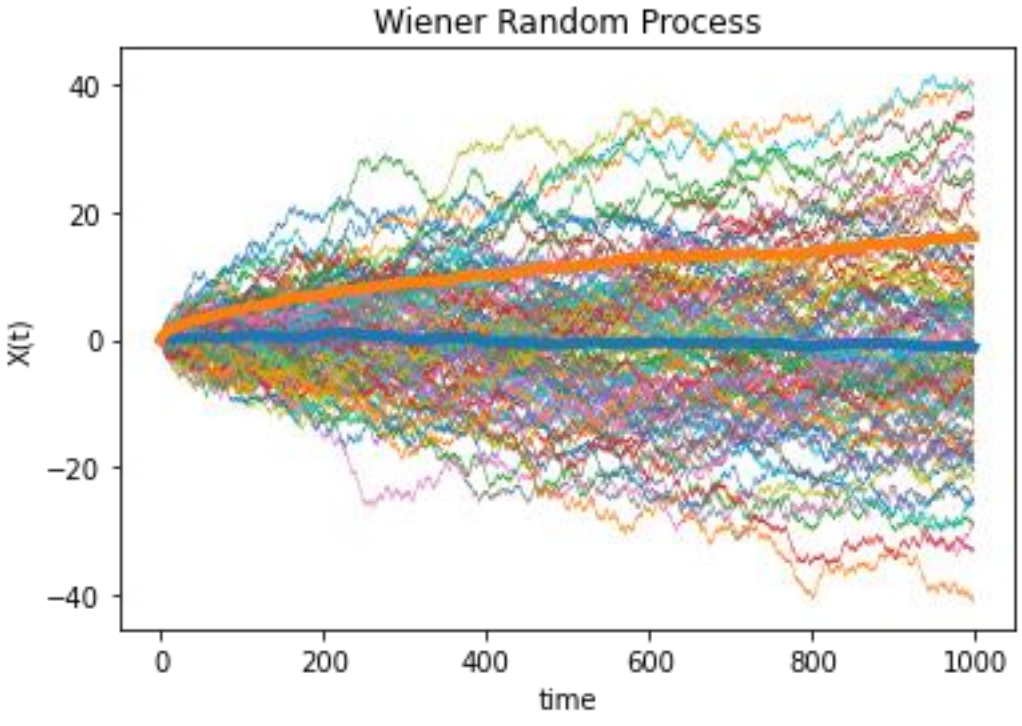
```
import random
import matplotlib.pyplot as plt
import numpy as np
from typing import List


def wiener_process(alpha: float, h: float, time: float)->np.ndarray:
    N: int = int(time//h)
    X_h = np.zeros(N, dtype="float64")
    for n in range(N-1):
        X_h[n+1] = X_h[n] + (alpha*h)**0.5 * random.randrange(-1, 2, 2)
# (-1, 1)
    return X_h

def mean(X_2D: np.ndarray, h: float, time: float)->np.ndarray:
    N: int = int(time//h)
    X_m = np.zeros(N, dtype="float64")
    X_2D = np.transpose(X_2D)

    for t in range(N):
        X_m[t] = np.mean(X_2D[t])
    return X_m

def variance(X_2D: np.ndarray, h: float, time: float)->np.ndarray:
    N: int = int(time//h)
    X_var = np.zeros(N, dtype="float64")
    X_2D = np.transpose(X_2D)

    for t in range(N):
        X_var[t] = np.var(X_2D[t])
    return X_var


alpha = 2
h = 0.1
time = 100
```

```
sample = 100

X_samples = np.empty((0, int(time//h)), dtype="float64") # N =
floor(time/h)
for _ in range(sample):
    X: np.ndarray = wiener_process(alpha=alpha, h=h, time=time) #1D
ndarray
    X_samples = np.append(X_samples, [X], axis=0) #2D ndarray

sample_mean = mean(X_2D=X_samples, h=h, time=time)
sample_var = variance(X_2D=X_samples, h=h, time=time)

for s in range(sample):
    plt.plot(X_samples[s], linewidth="0.5")
plt.plot(sample_mean, linewidth="4.0")
plt.plot(np.sqrt(sample_var), linewidth="4.0")
plt.title("Wiener Random Process")
plt.xlabel("time")
plt.ylabel("X(t)")
plt.show()
```

3. 결과

위의 코드를 통해 $\alpha = 2,\ h = 0.1,\ T = 100$의 constraints로 100개의 sample simulation을 진행한 결과 다음의 그래프를 얻을 수 있었다.

이때, variance의 square root와 sample mean을 각각 굵은 주황색, 파란색 선으로 표시하였다.



이로 인해 h가 0에 가까워지고 sample의 개수가 많아질수록 $E[X(t)] = 0$, $\sqrt{Var[X(t)]} = \sqrt{\alpha t}$ 에 수렴함을 확인할 수 있었다.