Stochastic Calculus

https://github.com/HST0077/FE2025

Wiener Process and Introduction to Stochastic Calculus

정규분포 확률밀도함수

Q. 확률변수 $X \sim N(2,2^2)$를 따른다고 할 때, 확률밀도함수(pdf)를 그려보아라.

```
from scipy.stats import norm
# cdf는 정규분포의 누적분포함수 F(x)
# cdf()에서 마지막 2개는 평균,표준편차임
print('X<=2 일 확률:',norm.cdf(2,2,2)) # F(2)
```

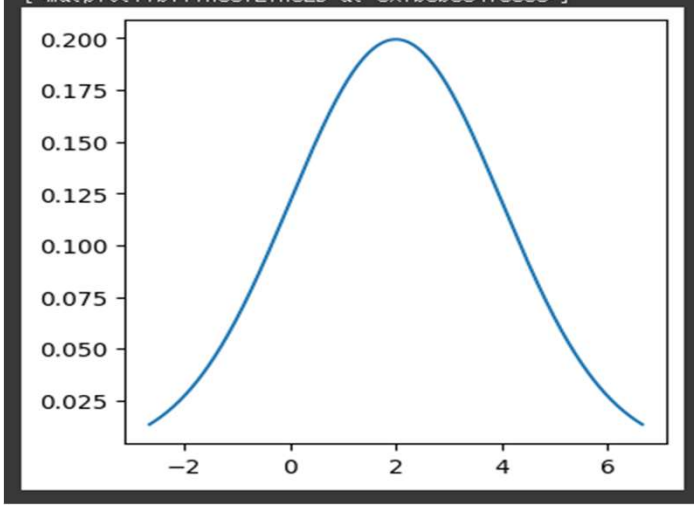
```
X<=2 일 확률: 0.5
```

```
# X~N(2,4), P(X<=k)=0.5 인 k를 구할 때
print('P(X<=k)=0.5을 만족하는 k:',norm.ppf(0.5,2,2))
```

```
P(X<=k)=0.5을 만족하는 k: 2.0
```

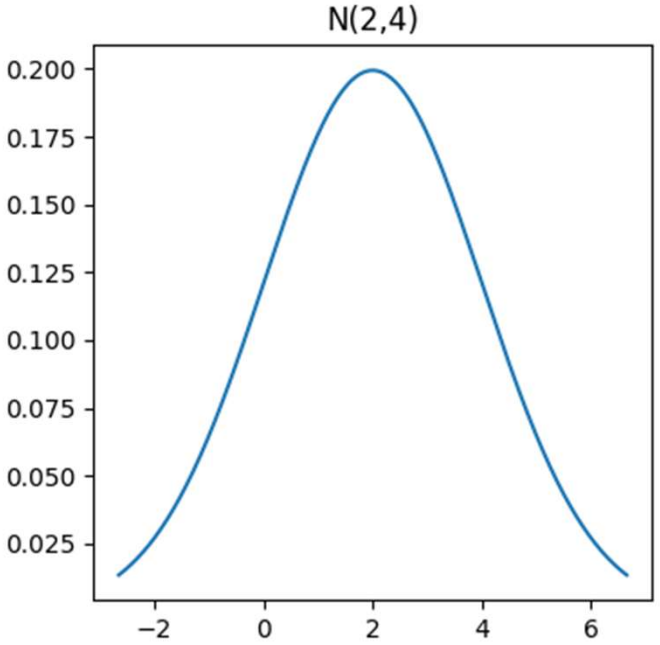
```
import matplotlib.pyplot as plt
import numpy as np
# 확률값 0.01에서 0.99까지 1000개로 나누기
x1=np.linspace(norm.ppf(0.01,2,2),norm.ppf(0.99,2,2),1000)
# pdf는 확률밀도함수 f(x)
plt.figure(figsize=(4,4))
plt.plot(x1,norm.pdf(x1,2,2))
```

```
[<matplotlib.lines.Line2D at 0x7b8b09473350>]
```



Random Variable vs Stochastic Process

Q. 확률변수 X가 시간에 따라서 변하는 정규분포라 한다면 이를 어떻게 표현할 수 있을까?



확률변수의 param이 변하면...

$$X \sim N(2t, 2t)$$

Comments

- 엄밀한 정의는 아니지만 시간이나 공간에 따라서 변하는 확률변수들을 표현하기 위해 등장한 개념이 확률과정 혹은 추계적과정(stochastic process)이라고 함
- 가장 대표적인 stochastic process 중 하나가 바로 Wiener process 혹은 Brownian Motion이라고 부르는 것인데, 정규분포의 평균과 분산이 시간의 함수로 주어짐
- 고등학교에서 배운 미적분(calculus)은 확률적으로 변하는 변수가 아닌 결정론적(deterministic) 변수를 대상으로 한 것인데, 변수들이 확률과정을 따르는 경우의 미적분을 확률미적분(stochastic calculus)라고 부름

정규분포 난수 생성

Q. 확률변수 $X \sim N(2,10^2)$를 따른다고 할 때, 정규분포 난수를 생성하고 생성된 난수를 바탕으로 올바른 확률변수 샘플들인지 검증해 보아라.

```
from scipy.stats import norm

# X~N(4,100)인 난수 100,000개 생성
D=norm.rvs(4,10,size=100000)

# 자료검증
a,b=norm.fit(D) # a,b는 각각 평균과 표준편차
# 평균과 분산 출력
print('평균:',a)
print('분산:',b**2)
```

```
평균: 3.9916582509777556
분산: 100.24157800523682
```

Deterministic Integration

Q. $f(x) = x^2$ 이라고 할 때, $\int_0^2 f(x)dx$의 값을 구하여 보아라.

```
# Sympy를 이용한 적분
from sympy import symbols, integrate

x=symbols('x', real=True)
integrate(x**2,(x,a,b))
```

$$\frac{8}{3}$$

```
# 위 셀의 결과 numeric
_.evalf()
```

```
2.66666666666667
```

```
# 구분구적법을 이용한 정적분 계산
import numpy as np

def f(x): # 함수 정의
    return x**2
n=1000 # 직사각형의 갯수
dx = (b - a) / n  # 각 직사각형의 가로 길이
x_samples = a + dx * np.arange(0, n) # 직사각형의 높이는 왼쪽모서리이용

# 높이 f(x_samples)을 구해서 넓이 = f(x_i)*dx 합
res=np.sum(f(x_samples) * dx)

print("구분구적법에 의한 적분:", res)
```

```
구분구적법에 의한 적분: 2.6626680000000005
```

Deterministic Integration Using MC simulation

Q. $f(x) = x^2$ 이라고 할 때, $\int_0^2 f(x)dx$의 값을 Monte Carlo Simulation을 이용하여 계산해 보아라.

```python
import numpy as np

# 1. [a,b] 균일 난수 생성
N=100000 # 시뮬레이션 회수
np.random.seed(10) # seed 고정
a,b=0,2 # 적분구간
xs = np.random.uniform(low=a, high=b, size=N)

# 2. f(x)=x^2 계산
fx = xs ** 2

# 3. 평균 * 구간길이
estimate = (b - a) * fx.mean()

# 4. 결과 출력
print("몬테카를로 추정값:", estimate)
```

```
몬테카를로 추정값: 2.6556377948675993
```

```python
# MC의 수렴성 확인
def monte_carlo_integral_x2(a=0.0, b=2.0, N=10_000, seed=None):

    xs = np.random.uniform(low=a, high=b, size=N)
    fx = xs ** 2
    estimate = (b - a) * fx.mean()
    return estimate

# 여러 번 돌려서 분포도 확인
trials = 20
ests = [monte_carlo_integral_x2(N=100_000) for _ in range(trials)]
print("20번 반복했을 때 평균:", np.mean(ests))
print("20번 반복했을 때 표준편차:", np.std(ests))
```

```
20번 반복했을 때 평균: 2.668152276890315
20번 반복했을 때 표준편차: 0.006066426929479602
```

Stochastic Integration, $\int_0^T dW_t$

Q. ΔW라는 확률변수는 평균이 0이고 분산이 Δ인 정규분포를 따른다고 하자. ΔW~$N(0,\Delta)$라고 표현하며, 여기서 Δ는 아주 작은 시간 간격이고, ΔW들은 서로 독립이라고 가정하자. 이 때, $\int_0^1 dW_t$의 값을 구하여 보아라.

$$\int_0^1 dW_t \cong \lim_{N\to\infty} \sum_{i=1}^{N} dW_i \,, dW_i \sim N(0,d)$$

	Sim_1	⋯	Sim_M
Interval_1	dW	⋯	dW
⋮	⋮	⋱	⋮
Interval_N	dW	⋯	dW
칼럼합	$\int_0^1 dW_t \_1$		$\int_0^1 dW_t \_N$

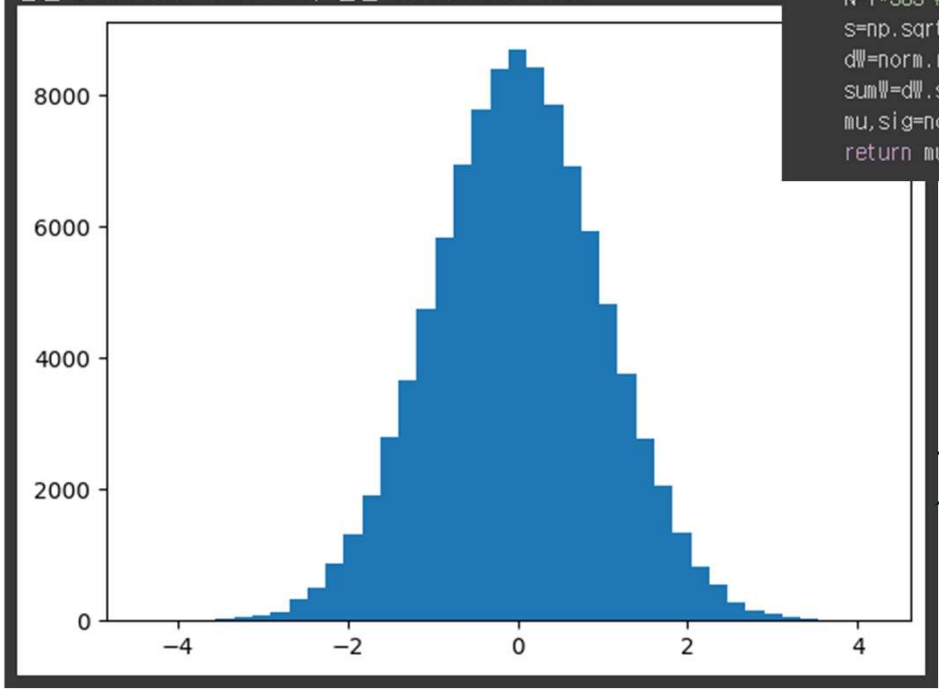
```python
import numpy as np
from scipy.stats import norm

sim=100000
N=300
s=np.sqrt(1/N) # d=1/N, d는 직사각형의 가로길이
# N(0,s)를 따르는 [N, sim] 크기의 정규난수 생성
dW=norm.rvs(0,s,size=[N,sim])
sumW=dW.sum(axis=0) # 각 열별로 누적합을 구함
mu,sig=norm.fit(sumW)
# 평균과 분산 출력
print('평균:',mu)
print('분산:',sig**2)
```

```
평균: 0.0024834914339944268
분산: 0.9955853839159997
```

Stochastic Integration, $\int_0^T dW_t$ (Cont'd)

```
import matplotlib.pyplot as plt

plt.hist(sumW,bins=40) # 히스토그램으로 그리기
mu,sig=norm.fit(sumW)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```

```
평균:0.0024834914339944268, 분산:0.9955853839159997
```

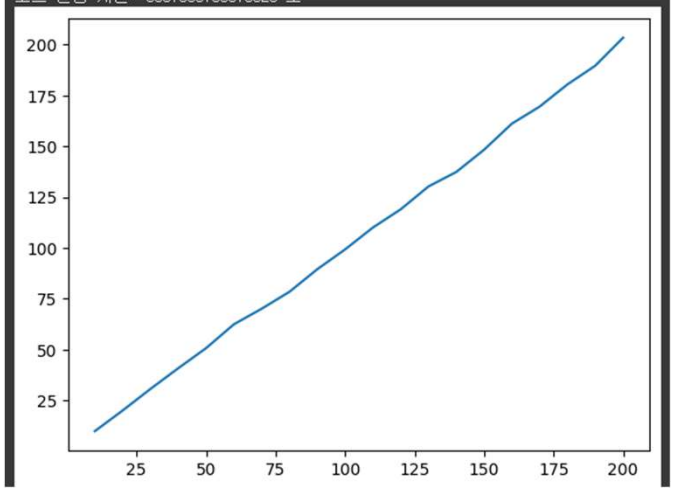



```
def BMsum(T,sim):
    import numpy as np
    from scipy.stats import norm
    import matplotlib.pyplot as plt
    # T: maturity in years
    # sim: 시뮬레이션 횟수
    N=T*365 # 일 단위로 환산한 만기까지의 기간
    s=np.sqrt(1/365) # 하루를 년으로 환산하여 표준편차화
    dW=norm.rvs(0,s,[N,sim]) # [N,sim] 크기 정규난수 생성
    sumW=dW.sum(axis=0) # 각 열별로 누적합을 구함
    mu,sig=norm.fit(sumW)
    return mu, sig**2
```

$$\int_0^T dW_t \sim N(0,T)$$

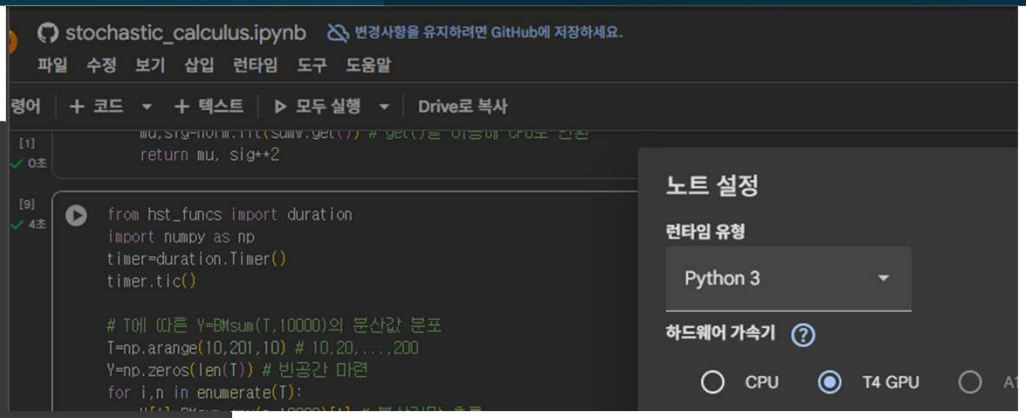
```
from hst_funcs import duration
timer=duration.Timer()
timer.tic()
# T에 따른 Y=BMsum(T,10000)의 분산값 분포
T=np.arange(10,201,10) # 10,20,...,200
Y=np.zeros(len(T)) # 빈공간 마련
for i,n in enumerate(T):
    Y[i]=BMsum(n,10000)[1] # 분산값만 추출
# 분포그리기
plt.plot(T,Y)
timer.toc()
```

```
코드 실행 시간: 305.586138010025 초
```



MC simulation using GPU

```python
# GPU 이용
def BMsum_gpu(T,sim):
    import cupy as cp
    from scipy.stats import norm
    import matplotlib.pyplot as plt
    import numpy as np
    # T: maturity in years
    # sim: 시뮬레이션 횟수
    N=T*365 # 일 단위로 환산한 만기까지의 기간
    s=cp.sqrt(1/365) # 하루를 년으로 환산하여 표준편차화
    dW=cp.random.normal(0,s,[N,sim]) # [N,sim] 크기 정규난수 생성
    sumW=cp.sum(dW,axis=0) # 각 열별로 누적합을 구함
    mu,sig=norm.fit(sumW.get()) # get()을 이용해 CPU로 변환
    return mu, sig**2
```



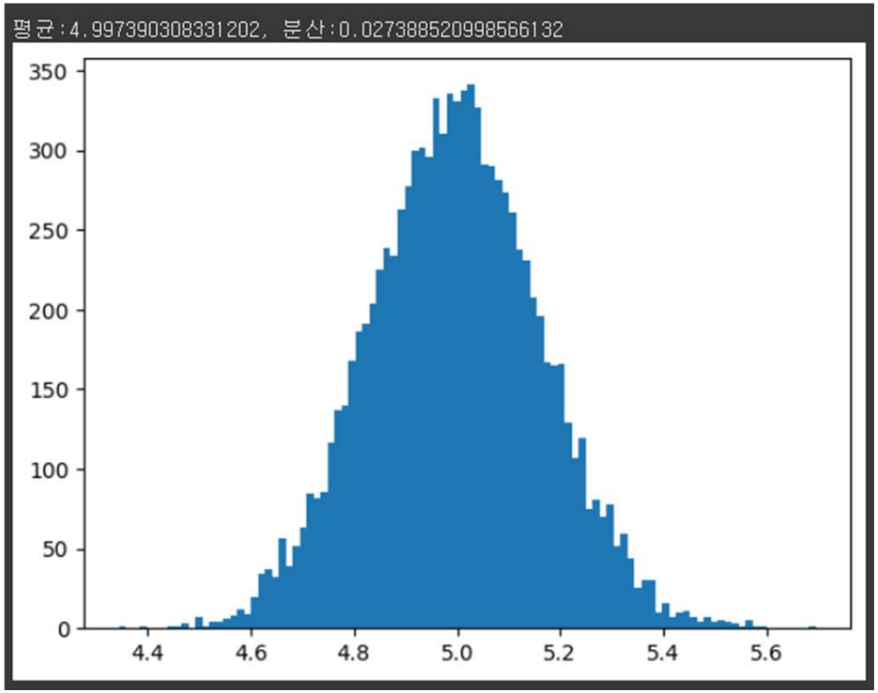
코드 실행 시간: 244.66809058189392 초

코드 실행 시간: 7.78387451171875 초

Stochastic Integration, $\int_0^T (dW_t)^2$

Q. $\Delta W \sim N(0, \Delta)$일 때, $\int_0^5 (dW_t)^2$의 값을 구하여 보아라.

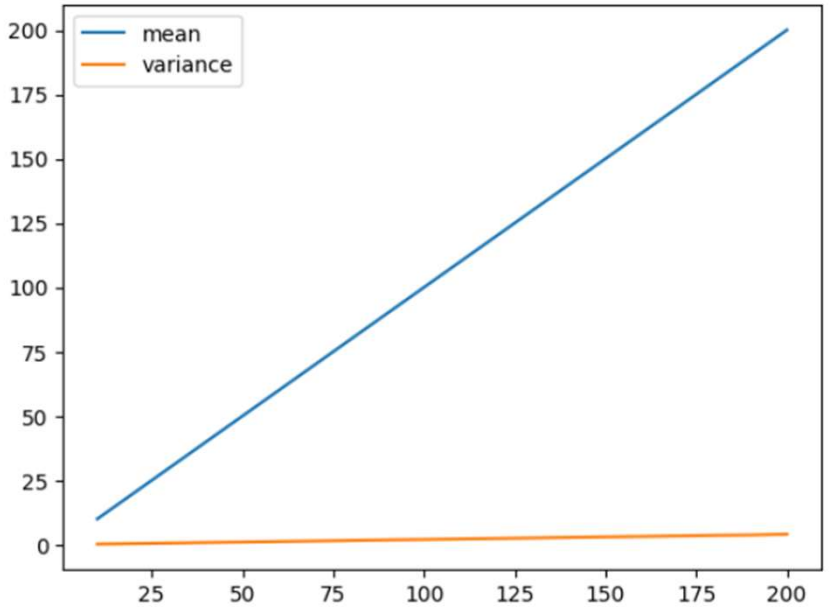
```
import cupy as cp
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

T=5 # num of years
N=T*365 # 일 단위로 환산한 만기까지의 기간
sim=10000 # simulation 회수
s=cp.sqrt(1/365) # 하루를 년으로 환산하여 표준편차화
dW=cp.random.normal(0,s,[N,sim]) # [N,sim] 크기 정규난수 생성
dW2=dW**2
sumW=cp.sum(dW2,axis=0) # 각 열별로 누적합을 구함
plt.hist(sumW.get(),bins=100)
mu,sig=norm.fit(sumW.get()) # get()을 이용해 CPU로 변환
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```



Stochastic Integration, $\int_0^T (dW_t)^2$ (Cont'd)

```
# 함수로 표현
def BMsum2_gpu(T,sim,N_interval):
    import cupy as cp
    from scipy.stats import norm
    N=T*N_interval
    sim=30000 # simulation 회수
    s=cp.sqrt(1/N_interval) # 하루를 년으로 환산하여 표준편차화
    dW=cp.random.normal(0,s,[N,sim]) # [N,sim] 크기 정규난수 생성
    dW2=dW**2
    sumW=cp.sum(dW2,axis=0) # 각 열별로 누적합을 구함
    mu,sig=norm.fit(sumW.get()) # get()을 이용해 CPU로 변환
    return mu, sig**2
```

```
# T에 따른 그래프
T=np.arange(10,201,10) # 10,20,...,200
M,S=np.zeros(len(T)),np.zeros(len(T))
for i,t in enumerate(T):
    M[i],S[i]=BMsum2_gpu(t,10000,100)
# 분포그리기
plt.plot(T,M,label='mean')
plt.plot(T,S,label='variance')
```

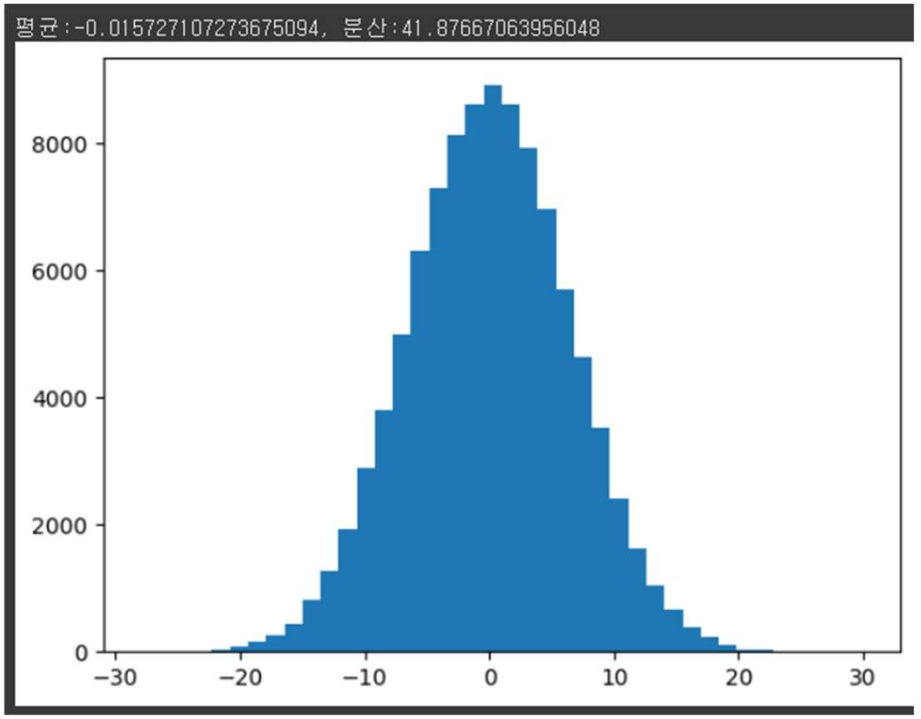


$$\int_0^T (dW_t)^2 = T \rightarrow (dW)^2 \cong dt$$

Stochastic Integration, $\int_a^b kdW_t$

Q. $\Delta W \sim N(0,\Delta)$일 때, $\int_1^5 dW_t$의 값을 구하여 보아라.

```
import cupy as cp
from scipy.stats import norm
a,b=0,5
T=b-a
sim,N_interval=100000,500
N=T*N_interval
s=cp.sqrt(1/N_interval)
dW=cp.random.normal(0,s,[N,sim])
# t를 (2500, 1) 형태로 변환
# (2500,) → (2500, 1)
t = cp.linspace(0, T, N).reshape(-1, 1)
integrand=t*dW
sumW=cp.sum(integrand,axis=0)
mu,sig=norm.fit(sumW.get())
plt.hist(sumW.get(),bins=40)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```



$$\int_a^b kdW_t \sim N(0, k^2(b-a))$$

참고: cumsum, tile

```
# cumsum 의 이해

import numpy as np

D=np.random.randint(10,size=[4,5])
D_row=np.cumsum(D,axis=0) # 행방향 누적합
D_col=np.cumsum(D,axis=1) # 열방향 누적합
D,D_row,D_col
```

```
(array([[3, 3, 9, 4, 3],
        [1, 5, 2, 1, 2],
        [3, 4, 4, 4, 6],
        [3, 1, 9, 2, 1]]),
 array([[ 3,  3,  9,  4,  3],
        [ 4,  8, 11,  5,  5],
        [ 7, 12, 15,  9, 11],
        [10, 13, 24, 11, 12]]),
 array([[ 3,  6, 15, 19, 22],
        [ 1,  6,  8,  9, 11],
        [ 3,  7, 11, 15, 21],
        [ 3,  4, 13, 15, 16]]))
```

```
# tile의 이해
D1=np.random.randint(10,size=[2,3])

# D1의 모양을 첫번째 축으로 3번
# 두번째 축방향으로 4번 반복함
D1,np.tile(D1,(3,4))
```

```
(array([[4, 2, 6],
        [9, 3, 3]]),
 array([[4, 2, 6, 4, 2, 6, 4, 2, 6, 4, 2, 6],
        [9, 3, 3, 9, 3, 3, 9, 3, 3, 9, 3, 3],
        [4, 2, 6, 4, 2, 6, 4, 2, 6, 4, 2, 6],
        [9, 3, 3, 9, 3, 3, 9, 3, 3, 9, 3, 3],
        [4, 2, 6, 4, 2, 6, 4, 2, 6, 4, 2, 6],
        [9, 3, 3, 9, 3, 3, 9, 3, 3, 9, 3, 3]]))
```

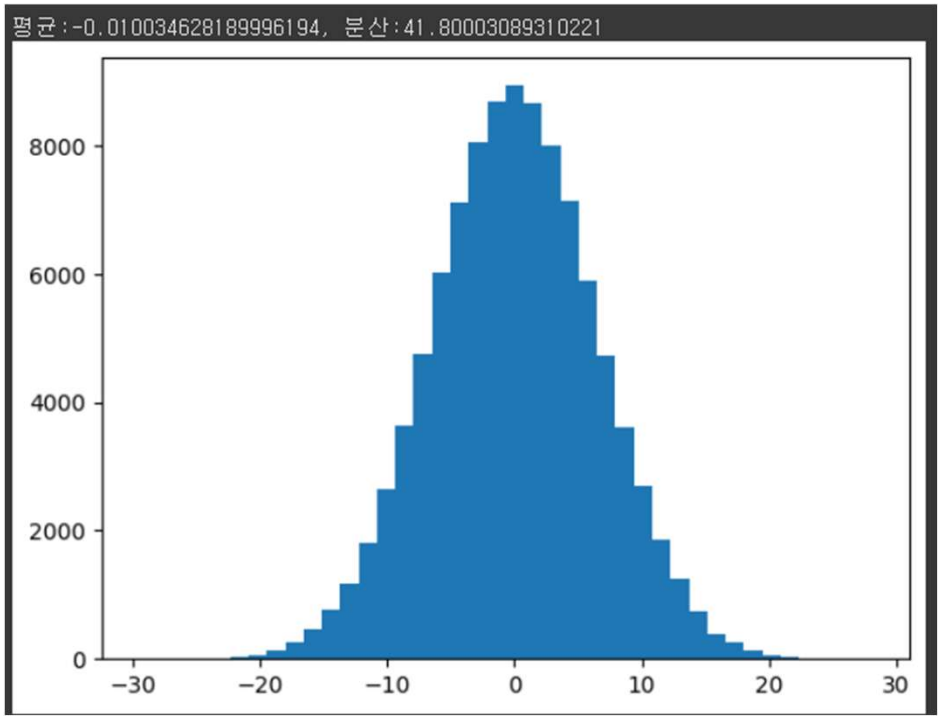
```
# tile은 BlockMatrix 개념
from sympy import MatrixSymbol
from sympy import Matrix,BlockMatrix

D1=MatrixSymbol('D1',4,5)
D2=[[D1] * 4] * 3
BlockMatrix(D2)
```

$$\begin{bmatrix} D_1 & D_1 & D_1 & D_1 \\ D_1 & D_1 & D_1 & D_1 \\ D_1 & D_1 & D_1 & D_1 \end{bmatrix}$$

Stochastic Integration, $\int_a^b tdW_t$

Q. $\Delta W \sim N(0,\Delta)$일 때, $\int_0^5 tdW_t$의 값을 구하여 보아라.

```
# 시간행렬로 변환하여 재코딩
a,b=0,5
T=b-a
sim,N_interval=100000,500
N=T*N_interval
s=cp.sqrt(1/N_interval)
dW=cp.random.normal(0,s,[N,sim])

# 시간간격 h를 기준으로 dW와 동일형태의 행렬로 변환
h=(b-a)/N # 시간간격
h_col=cp.array([h]*N) # 동일간격 시간축
t_col=cp.cumsum(h_col,axis=0) # ih의 값 생성
t_col=t_col.reshape(-1,1) # 칼럼 행렬로 변환
t_mat=cp.tile(t_col,(1,sim)) # sim 개수만큼 행렬복제
# 두행렬의 곱(product)이 아니라, Hadamard Product 수행
integrand=t_mat*dW

sumW=cp.sum(integrand,axis=0)
mu,sig=norm.fit(sumW.get())
plt.hist(sumW.get(),bins=40)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```



Ito lemma

Ito Lemma

$$(dW)^2 \cong dt$$

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t, \qquad f(X_t, t)$$

$$df = \frac{\partial f}{\partial X}dX + \frac{\partial f}{\partial t}dt + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}(dX)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial t^2}(dt)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial X \partial t}(dX)(dt) + \cdots$$

$$df = \left(\frac{\partial f}{\partial t} + a\frac{\partial f}{\partial X} + \frac{1}{2}b\frac{\partial^2 f}{\partial X^2}\right)dt + b\frac{\partial f}{\partial X}dW$$

Stochastic Differential Equations

Q. $dS_t = rS_t dt + \sigma S_t dW_t$ 일 때, Ito lemma를 이용하여 $d\ln S_t$ 를 간단히 표현해 보아라.

$$d\ln S_t = \frac{1}{S_t}(dS_t) - \frac{1}{2}\frac{1}{(S_t)^2}(dS_t)^2 + \cdots$$

$$\cong \frac{1}{S_t}(rS_t dt + \sigma S_t dW_t) - \frac{1}{2}\frac{1}{(S_t)^2}(\sigma^2 S_t^2 dt)$$

$$= \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t$$

lto integration

Stochastic Integration, $\int_a^b W_t dW_t$

Q. $\Delta W \sim N(0, \Delta)$일 때, $\int_0^5 W_t dW_t$의 값을 구하여 보아라.

$$h = \frac{5-0}{N}, \int_0^5 t dt = \lim_{N\to\infty} \sum_{i=1}^{N} (ih)h$$

일반적 정적분문제 살펴보기, $\int_0^5 t\,dt$

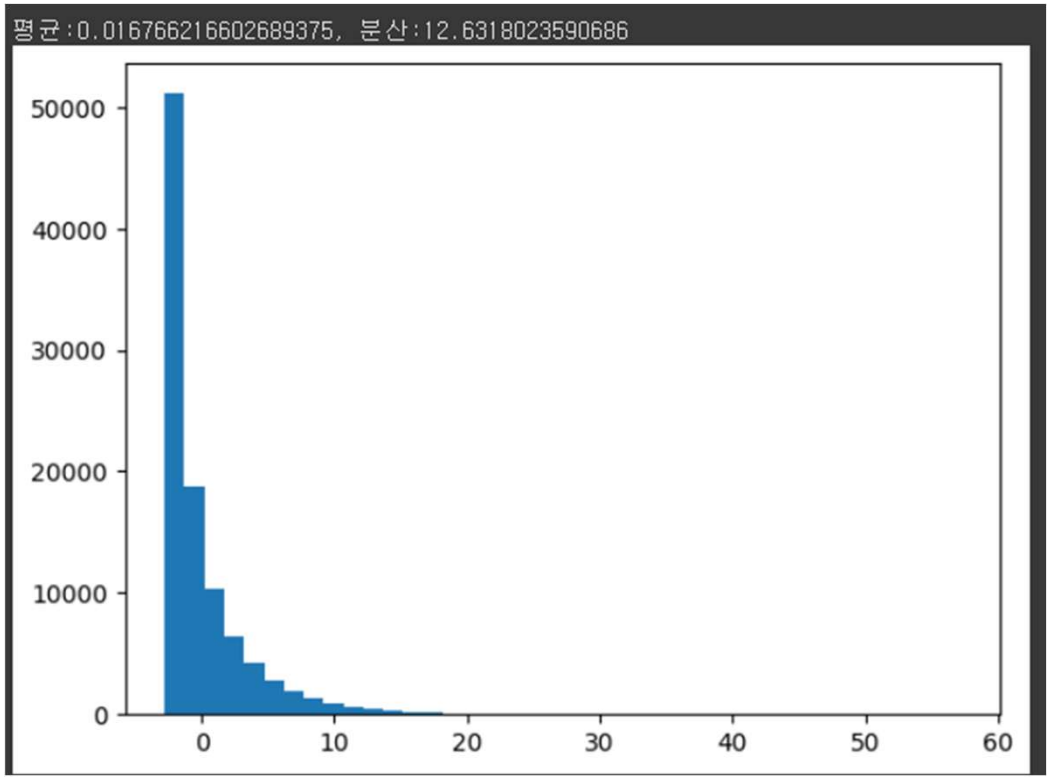
```
# deterministic integration
T,N=5,100000
h=T/N
# N+1개의 시점 생성
t_ax=np.linspace(0,T,N+1)

# 직사각형의 왼쪽 함수값을 높이로 구함
left_sum=np.sum(t_ax[:-1]*h)
print('왼쪽을 기준으로:',left_sum)

# 직사각형의 오른쪽 함수값을 높이로 구함
right_sum=np.sum(t_ax[1:]*h)
print('오른쪽을 기준으로:',right_sum)

# 직사각형의 중앙값을 높이로 구함
center_sum=np.sum((t_ax[:-1]+t_ax[1:])/2*h)
print('중앙값을 기준으로:',center_sum)
```

```
왼쪽을 기준으로: 12.499875000000003
오른쪽을 기준으로: 12.500125
중앙값을 기준으로: 12.500000000000002
```

Stochastic Integration, $\int_a^b W_t dW_t$ (Cont'd)

Ito Integration

```
# N(0,s)을 따르는 [N, sim] 크기의 정규난수 생성
dW=cp.random.normal(0,s,[N,sim])

# Wt 행렬 만들기
W=cp.cumsum(dW,axis=0)
# 첫 행에 zero vector 추가
zero_row=cp.zeros((1,W.shape[1]))
#(N+1,sim) 행렬로 변환
W=cp.vstack([zero_row,W])

# 사각형의 왼쪽에서 높이를 구함
integrand=W[:-1]*dW # Hadamard 곱

sumW=cp.sum(integrand,axis=0)
mu,sig=norm.fit(sumW.get())
plt.hist(sumW.get(),bins=40)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```



Stochastic Integration, $\int_a^b W_t dW_t$ (Cont'd)

Stratonovich Integration

```
# Stratonovich integral
# 중앙값에서 높이를 구함
integrand=(W[1:]+W[:-1])/2*dW

sumW=cp.sum(integrand,axis=0)
mu,sig=norm.fit(sumW.get())
plt.hist(sumW.get(),bins=40)
```

Skorokhod Integration

```
# Skorokhod integration
# 사각형의 오른쪽에서 높이를 구함
integrand=W[1:]*dW # Hadamard 곱

sumW=cp.sum(integrand,axis=0)
mu,sig=norm.fit(sumW.get())
plt.hist(sumW.get(),bins=40)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```

```
평균:5.016825593905128, 분산:12.653586794531922
```

Stochastic Integration, $\int_a^b W_t dW_t$ (Cont'd)

Using Ito lemma

$$d(W_t)^2 = 2W_t(dW_t) + \frac{1}{2}2(dW_t)^2 + \cdots$$

$$\cong 2W_t(dW_t) + \frac{1}{2}2dt = 2W_t dW_t + dt$$

$$\int_a^b d(W_t)^2 = 2\int_a^b W_t dW_t + \int_a^b dt$$

$$W_b^2 - W_a^2 = 2\int_a^b W_t dW_t + (b-a)$$

$$W_5^2 = 2\int_0^5 W_t dW_t - 5$$

```
# Ito integral 수식에 의한 값
# W행렬의 마지막행에 W_T 값들이 존재함
W_T=cp.asnumpy(W[-1,:])
T=b-a
mu, sig = norm.fit(0.5*(W_T**2-T))
print('평균:',mu, '분산:',sig**2)
```

```
평균: 0.0167959052539082 분산: 12.637652728777619
```

Ito Integration result

```
평균:0.016766216602689375, 분산:12.6318023590686
```

Solving Stochastic Differential Equations

SDE #1

Q. $dY_t = Y_t dW_t$ 때, Y_t를 구하여라. (단, $Y_0 = 1, W_0 = 0$)

$$d\ln Y_t = \frac{1}{Y_t}(dY_t) - \frac{1}{2}\frac{1}{(Y_t)^2}(dY_t)^2 + \cdots \cong dW_t - \frac{1}{2}dt$$

$$\int_0^t d\ln Y_s = \int_0^t dW_s - \frac{1}{2}\int_0^t ds$$

$$\ln Y_t - \ln Y_0 = W_t - W_0 - \frac{1}{2}t$$

$$\therefore Y_t = e^{W_t - \frac{1}{2}t}$$

```
# N(0,s)을 따르는 [N, sim] 크기의 정규난수 생성
dW=cp.random.normal(0,s,[N,sim])
# Wt 행렬 만들기
W=cp.cumsum(dW,axis=0)
# 첫 행에 zero vector 추가
zero_row=cp.zeros((1,W.shape[1]))
#(N+1,sim) 행렬로 변환
W=cp.vstack([zero_row,W])

Y=cp.exp(W-0.5*t_mat)
mu,sig=norm.fit(Y[-1,:].get())
plt.hist(sumW.get(),bins=40)
print('평균:{0}, 분산:{1}'.format(mu,sig**2))
```

```
평균:0.9836627528299849, 분산:58.800162932199456
```

SDE #2: Geometric Brownian Motion

Q. $dN_t = rN_t dt + \sigma N_t dW_t$ 일 때, N_t를 구하여라. (단, $W_0 = 0$)

$$d\ln N_t = \frac{1}{N_t}(dN_t) - \frac{1}{2}\frac{1}{(N_t)^2}(dN_t)^2 + \cdots \cong rdt + \sigma dW_t - \frac{1}{2}\sigma^2 dt = \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t$$

$$\int_0^t d\ln N_s = \left(r - \frac{1}{2}\sigma^2\right)t + \sigma\int_0^t dW_s$$

$$\therefore N_t = N_0 e^{\left(r-\frac{1}{2}\sigma^2\right)t+\sigma W_t}$$

Stock Price Path

Q. Random Walk을 따르는 주가의 path를 시뮬레이션 하여라.

$$d\ln(N_t) = \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t$$

$$\ln(N_{t+\Delta}) - \ln(N_t) = \left(r - \frac{1}{2}\sigma^2\right)\Delta + \sigma dW_t$$

$$\therefore \boldsymbol{N_{t+\Delta} = N_t e^{\left(r-\frac{1}{2}\sigma^2\right)\Delta+\sigma dW_t}}$$

```python
# 주가 시뮬레이션
def stockprices(N_0, T, r,sig, N_path):
    import numpy as np

    h=T/N_path
    s=np.sqrt(h) # 시간간격의 제곱근
    dW = np.random.normal(0, s, N_path)

    dlnN =h*(r-0.5*sig**2)+sig*dW
    N=N_0*np.exp(np.cumsum(dlnN))

    # 시작점을 N_0로 추가
    N = np.insert(N,0,N_0)

    return N
```

```python
plt.figure(figsize=(4,4))
plt.plot(stockprices(100,1,0.05,0.3,365))
plt.show()
```