HW7

May 5, 2023

1 Exercise Set 7

1.1 Mohaddeseh Mozaffari

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

2 Random walk:

3 A:

A: $x(N\Delta t) = s_1 + s_2 + \cdots + s_N = \sum_i s_i$

$\Delta x = 1/2 \rightsquigarrow$ mean of step in uniform dist.

$\langle x(N\Delta t)\rangle = \langle \sum_i s_i \rangle = \sum_i \langle s_i \rangle = N\langle s\rangle$

$\langle s\rangle = \int_0^1 ds\, s\, p(s) = \int_0^1 ds\, s\, [P_+ \delta(s-\Delta x) + P_- \delta(s+\Delta x)]$

$= \frac{1}{2}\left[\frac{1}{2} - \frac{1}{2}\right] = 0$

$\Rightarrow \langle x(N\Delta t)\rangle = 0$

$\langle s^2\rangle = \int_0^1 ds\, s^2\, [P_+ \delta(s-\Delta x) + P_- \delta(s+\Delta x)] = \frac{1}{2}\left[\frac{1}{4} + \frac{1}{4}\right] = \frac{1}{4}$

$\Rightarrow \langle \sigma_x^2(N\Delta t)\rangle = \frac{N}{4}$

B: $\langle x(N\Delta t)\rangle = N\langle s\rangle = 0$ for all gaussian

$\langle \sigma_x^2(N\Delta t)\rangle = N\langle s^2\rangle = N\sigma^2$

C: $\langle s\rangle = \int s\, ds\, [P_+ p(s) - P_- p(s)] = 0.4\int_{-4}^{4} s\, ds\, \tanh(s) = 0$

$\langle \sigma_x^2(N\Delta t)\rangle = N\langle s^2\rangle = 6.072(N)$

$\langle s^2\rangle = 0.4\int_{-4}^{4} s \tanh s\, ds = 0.4 \times 15.18 = 6.072$

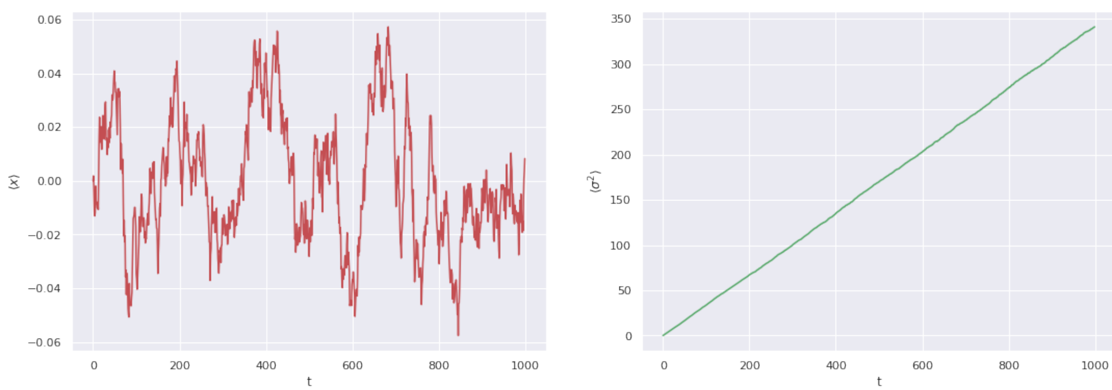
```
N = 10000
M = 1000
T = np.linspace(0,100,N)
dt = T[1] - T[0]
```

```
def RW_flat(p, N ,M):
    """random walk with step probability is flat

    Args:
        p (float): probability
        N (int): number of time steps
        M (int): number of ensemble

    Returns:
        2d_array: (M,N)
    """
    x = np.zeros((M,N))
```

```
    for ens in range(M):
        for t in range(1,N):
                r = np.random.random()
                dx = np.random.random()
                if r<= p:
                    x[ens,t] = x[ens,t-1] + dx
                else:
                    x[ens,t] = x[ens,t-1] - dx
    return x
```

[ ]:
```
X = RW_flat(0.5, 1000, 10000)
```

[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(18,6))

axes[0].plot(X.mean(axis=0), color="r")
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x \rangle$")

axes[1].plot(X.var(axis=0), color="g")
axes[1].set_xlabel("t")
axes[1].set_ylabel(r"$\langle \sigma^2 \rangle$")

plt.show()
```



4 B:

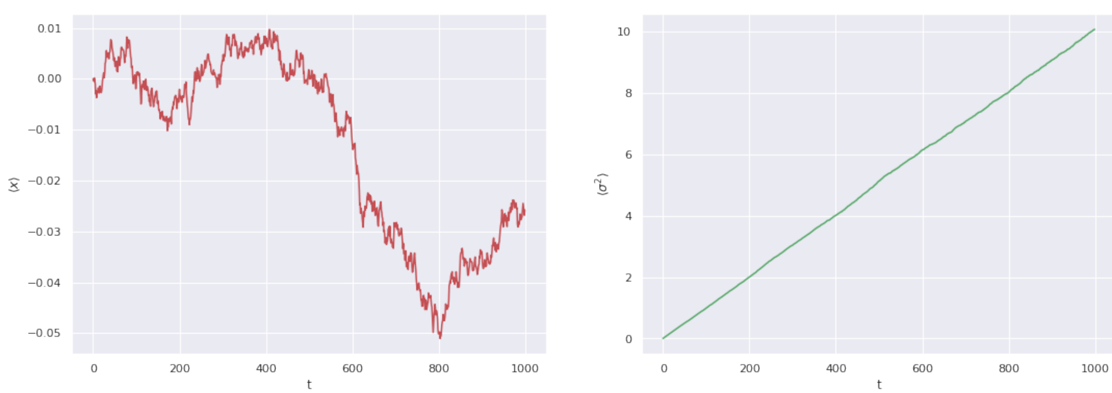
[ ]:
```
def RW_guassian(p, sigma, N ,M):
    """random walk with probability of stpe is guassian

    Args:
        p (float): probability
        sigma (float): standar deviation of guassian
```

```
        N (int): number of time steps
        M (int): number of ensemble

    Returns:
        2d_array: (M,N)
    """
    x = np.zeros((M,N))
    for ens in range(M):
        for t in range(1,N):
                r = np.random.random()
                dx = np.random.normal(0,sigma)
                if r<= p:
                    x[ens,t] = x[ens,t-1] + dx
                else:
                    x[ens,t] = x[ens,t-1] - dx
    return x
```

4.0.1 = 0.1

[ ]:
```
X1 = RW_guassian(0.5, 0.1, 1000, 10000)
```

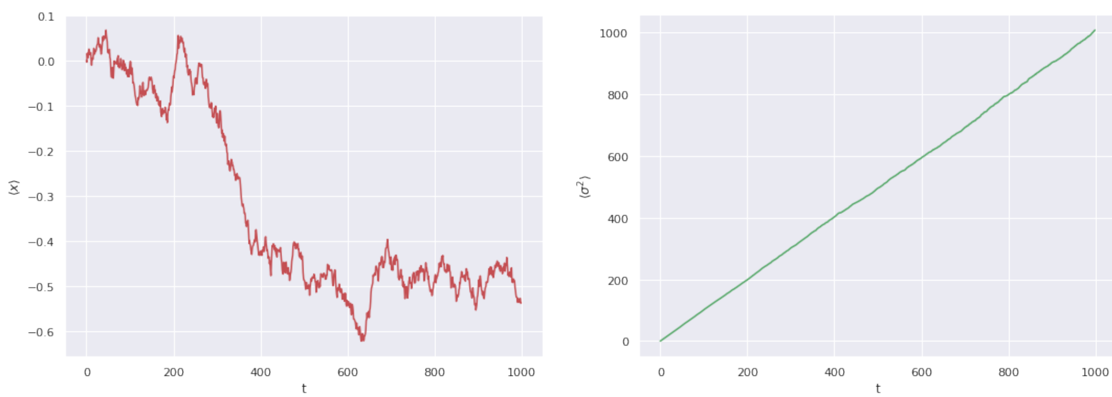
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(18,6))

axes[0].plot(X1.mean(axis=0), color="r")
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x \rangle$")

axes[1].plot(X1.var(axis=0), color="g")
axes[1].set_xlabel("t")
axes[1].set_ylabel(r"$\langle \sigma^2 \rangle$")

plt.show()
```



4.0.2 = 1

[ ]:
```python
X2 = RW_guassian(0.5, 1, 1000, 10000)
```

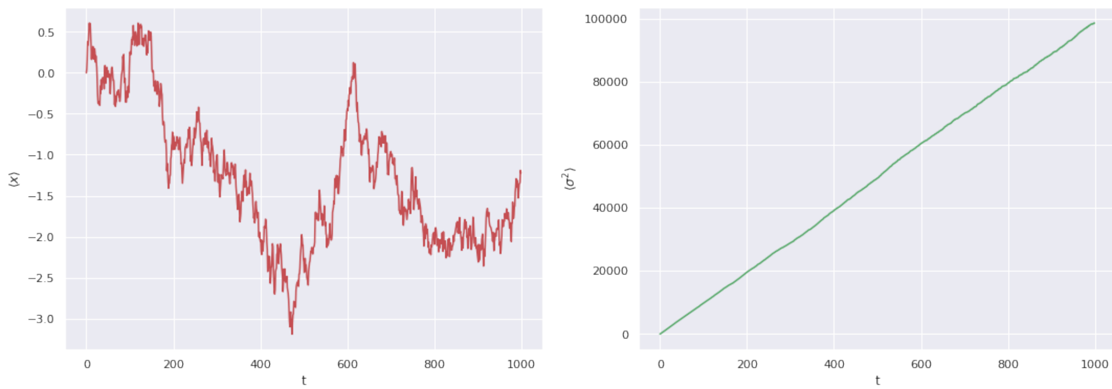
[ ]:
```python
fig, axes = plt.subplots(1, 2, figsize=(18,6))

axes[0].plot(X2.mean(axis=0), color="r")
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x \rangle$")

axes[1].plot(X2.var(axis=0), color="g")
axes[1].set_xlabel("t")
axes[1].set_ylabel(r"$\langle \sigma^2 \rangle$")

plt.show()
```



4.0.3 = 10

[ ]:
```python
X3 = RW_guassian(0.5, 10, 1000, 10000)
```

[ ]:
```python
fig, axes = plt.subplots(1, 2, figsize=(18,6))

axes[0].plot(X3.mean(axis=0), color="r")
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x \rangle$")

axes[1].plot(X3.var(axis=0), color="g")
axes[1].set_xlabel("t")
axes[1].set_ylabel(r"$\langle \sigma^2 \rangle$")

plt.show()
```



5 C:

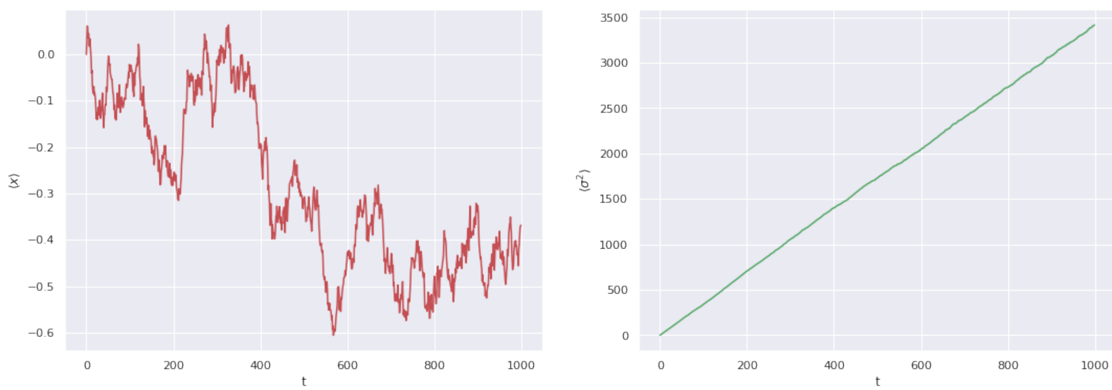
```
[ ]: def p_tanh(n):
         """random number from tanh(s)/s

         Args:
             n (int): number od random variable we want

         Returns:
             list: list of random variable
         """
         ps = []
         while len(ps) <= n:
             x = np.random.uniform(-4,4)
             y = np.random.uniform(0,1)
             yp = np.tanh(x)/x
             if y<= yp:
                 ps.append(x)
         return ps
```

```
[ ]: def RW_tanh(p, N ,M):
         """random walk with probability of stpe is tanh(s)/s

         Args:
             p (float): probability
             sigma (float): standar deviation of guassian
             N (int): number of time steps
             M (int): number of ensemble

         Returns:
             2d_array: (M,N)
         """
```

```
    x = np.zeros((M,N))
    for ens in range(M):
        for t in range(1,N):
                r = np.random.random()
                dx = p_tanh(1)[0]
                if r<= p:
                    x[ens,t] = x[ens,t-1] + dx
                else:
                    x[ens,t] = x[ens,t-1] - dx
    return x
```

[ ]:
```
Xt = RW_tanh(0.3, 1000, 10000)
```

[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(18,6))

axes[0].plot(Xt.mean(axis=0), color="r")
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x \rangle$")

axes[1].plot(Xt.var(axis=0), color="g")
axes[1].set_xlabel("t")
axes[1].set_ylabel(r"$\langle \sigma^2 \rangle$")

plt.show()
```

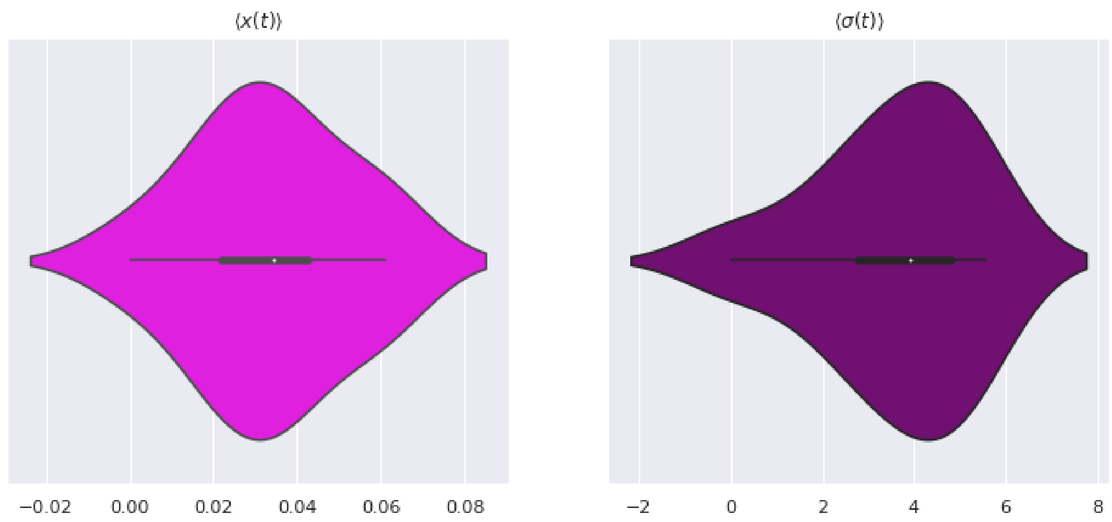


6 D:

6.1 t = 10

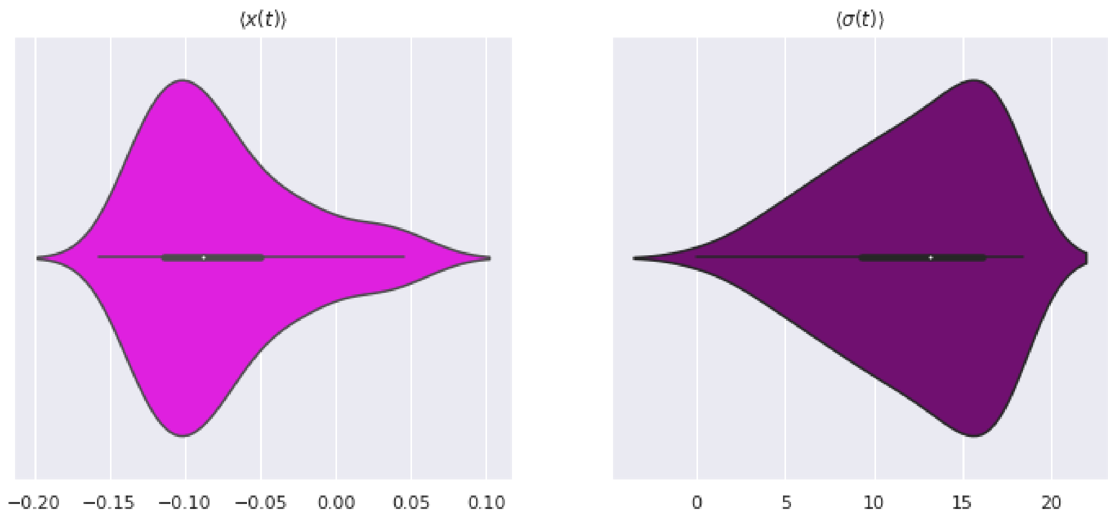
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))
sns.violinplot(x=Xt.mean(axis=0)[:10], ax=axes[0], color="magenta")
axes[0].set_title(r"$\langle x(t) \rangle$")
```

```
sns.violinplot(x=Xt.std(axis=0)[:10], ax=axes[1], color="purple")
axes[1].set_title(r"$\langle \sigma(t) \rangle$")

plt.show()
```



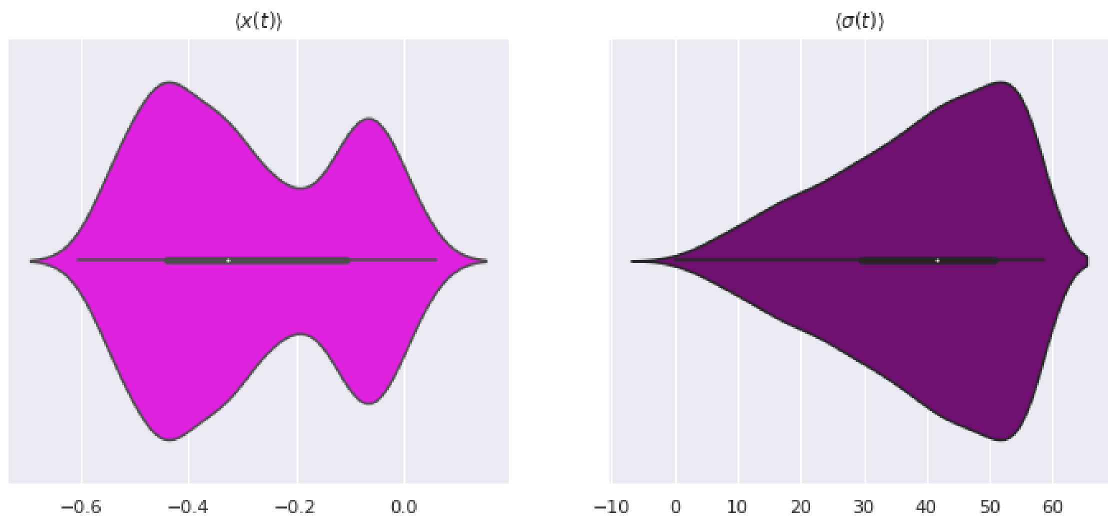
6.2 t = 100

[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))
sns.violinplot(x=Xt.mean(axis=0)[:100], ax=axes[0], color="magenta")
axes[0].set_title(r"$\langle x(t) \rangle$")

sns.violinplot(x=Xt.std(axis=0)[:100], ax=axes[1], color="purple")
axes[1].set_title(r"$\langle \sigma(t) \rangle$")

plt.show()
```



6.3 t = 1000

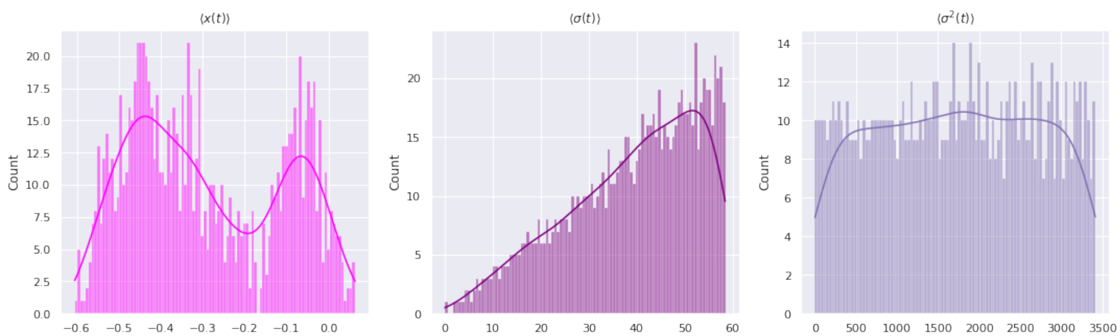
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))
sns.violinplot(x=Xt.mean(axis=0)[:1000], ax=axes[0], color="magenta")
axes[0].set_title(r"$\langle x(t) \rangle$")

sns.violinplot(x=Xt.std(axis=0)[:1000], ax=axes[1], color="purple")
axes[1].set_title(r"$\langle \sigma(t) \rangle$")

plt.show()
```



The violin plot shows the PDF, median, and 4 quartiles of data.

7 F:

```
[ ]: fig, axes = plt.subplots(1, 3, figsize=(18,5))

     sns.histplot(Xt.mean(axis=0), kde=True, bins=100, ax=axes[0], color="magenta")
     axes[0].set_title(r"$\langle x(t) \rangle$")

     sns.histplot(Xt.std(axis=0), kde=True, bins=100, ax=axes[1], color="purple")
     axes[1].set_title(r"$\langle \sigma(t) \rangle$")

     sns.histplot(Xt.var(axis=0), kde=True, bins=100, ax=axes[2], color="m")
     axes[2].set_title(r"$\langle \sigma^2(t) \rangle$")
     plt.show()
```



As we can see, The PDF of data is the same as the violin plots where we plotted in parts D and E.

8 Langevin particle:

```
[ ]: #set constant number
     g = 1
     ksi = 1
     v_0 = 0
     x_0 = 0
     N = 10000
     nens = 1000
```

```
[ ]: T = np.linspace(0,1000,N)
     dt = T[1] - T[0]
     V , X = [], []
```

```
[ ]: for e in range(nens):
         v = np.zeros(N)
         x = np.zeros(N)
         v[0] = v_0
```

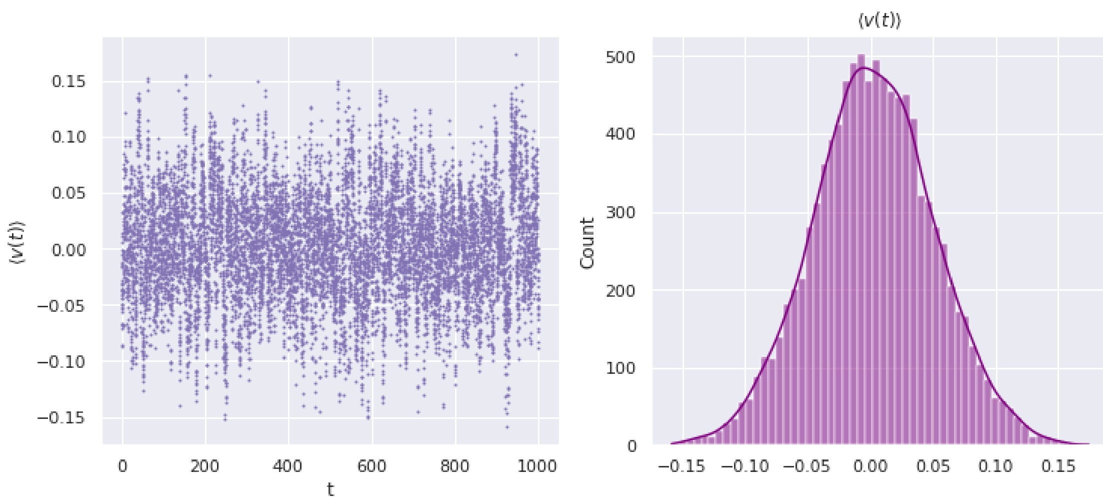
```
    x[0] = x_0
    for t in range(N):
        r1, r2 = np.random.random(2)
        eta = np.sqrt(-2*np.log(r1))*np.cos(2*np.pi*r2) * 2* g
        v[t] = v[t-1] - ksi*v[t-1]* dt + eta*(dt)**(0.5)
        x[t] = x[t-1] + v[t]*dt
    V.append(v)
    X.append(x)
```

[ ]:
```
V = np.array(V)
X = np.array(X)
```

8.1 A: $v(t)$

[ ]:
```
meanV = np.mean(V, axis=0)
```

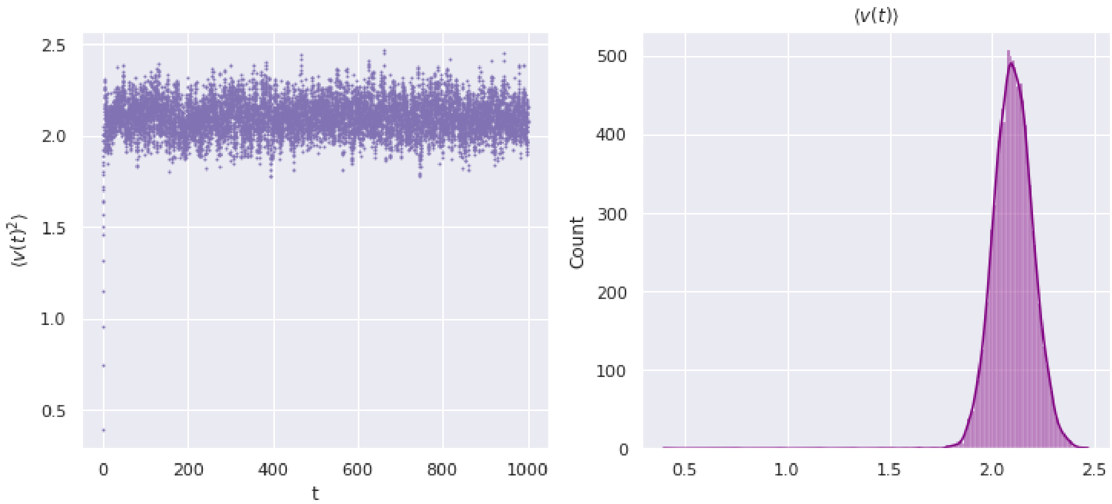
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))

axes[0].scatter(T, meanV, color="m",s=1)
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle v(t)\rangle$")

sns.histplot(meanV, kde=True, ax=axes[1], color="purple")
axes[1].set_title(r"$\langle v(t)\rangle$")
plt.show()
```



8.2 B: $v(t)^2$

[ ]:
```
meanV2 = np.mean(V**2, axis=0)
```

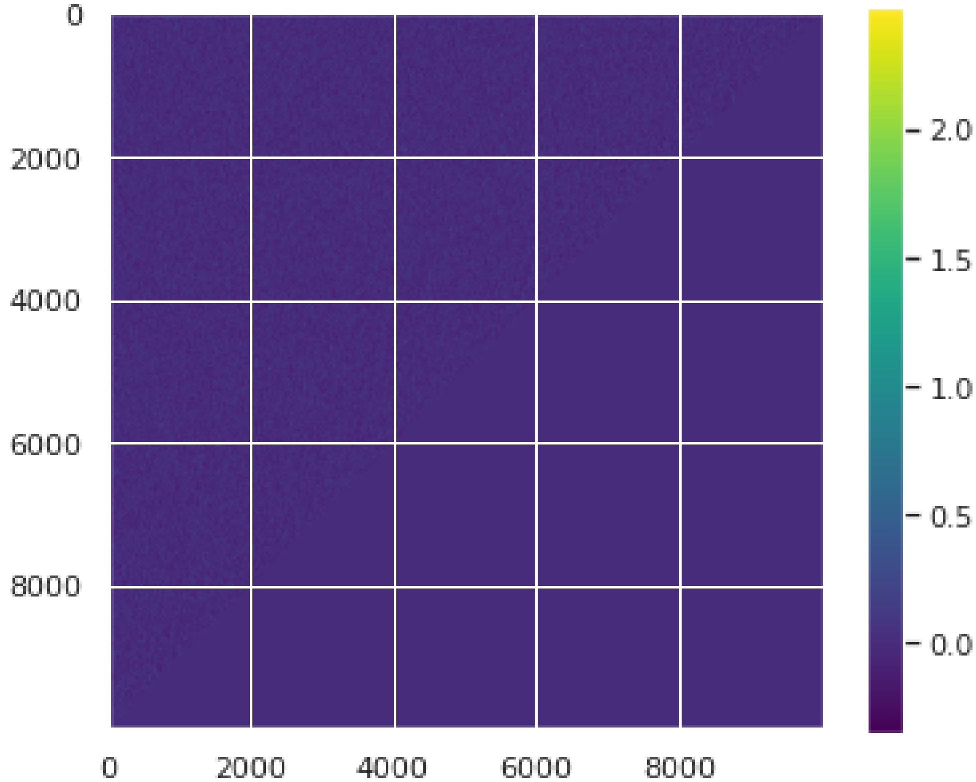
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))

axes[0].scatter(T, meanV2, color="m",s=1)
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle v(t)^2\rangle$")

sns.histplot(meanV2, kde=True, ax=axes[1], color="purple")
axes[1].set_title(r"$\langle v(t)\rangle$")
plt.show()
```



8.3 C: $v(t_1)v(t_2)$

[ ]:
```
Meanvv = np.zeros((N,N))
for tau in range(N):
    for t in range(N-tau):
        vv = np.mean(V[:,t]*V[:,t+tau])
        Meanvv[tau,t] = vv
```

[ ]:
```
plt.figure(figsize=(6,5))
plt.imshow(Meanvv, cmap="viridis")
plt.colorbar()
plt.show()
```



8.4 D: $x(t)$

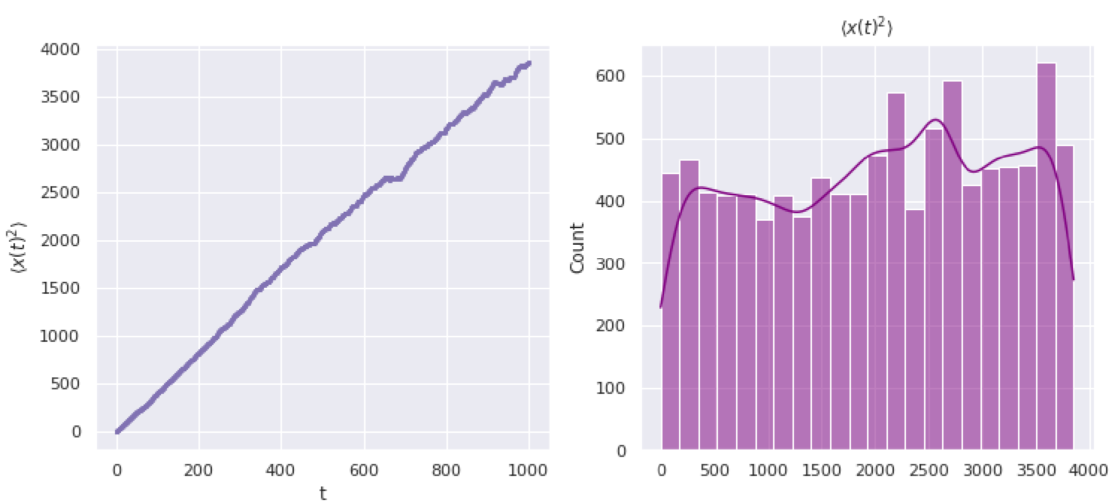
[ ]:
```
meanX= np.mean(X, axis=0)
```

[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))

axes[0].scatter(T, meanX, color="m",s=1)
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x(t)\rangle$")

sns.histplot(meanX, kde=True, ax=axes[1], color="purple")
axes[1].set_title(r"$\langle x(t)\rangle$")
plt.show()
```

8.5 E: $x(t)^2$

[ ]:
```
meanX2= np.mean(X**2, axis=0)
```

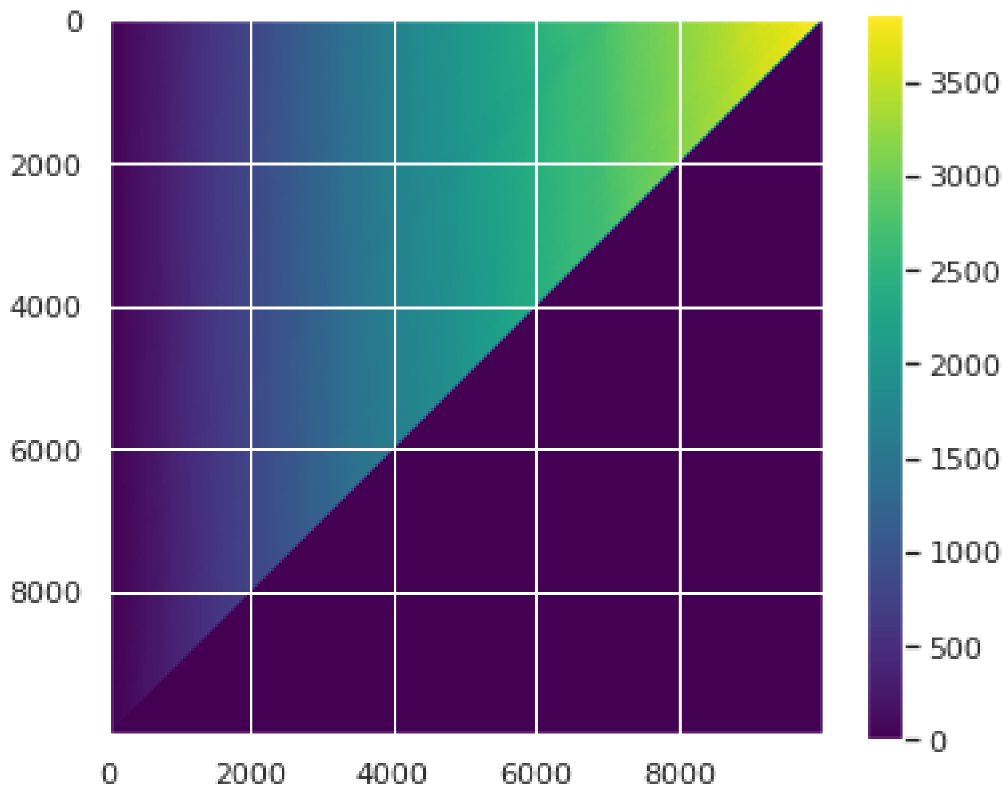
[ ]:
```
fig, axes = plt.subplots(1, 2, figsize=(12,5))

axes[0].scatter(T, meanX2, color="m",s=1)
axes[0].set_xlabel("t")
axes[0].set_ylabel(r"$\langle x(t)^2\rangle$")

sns.histplot(meanX2, kde=True, ax=axes[1], color="purple")
axes[1].set_title(r"$\langle x(t)^2\rangle$")
plt.show()
```



8.6 F: $x(t_1)x(t_2)$

[ ]:
```
Meanxx = np.zeros((N,N))
for tau in range(N):
    for t in range(N-tau):
        xx = np.mean(X[:,t]*X[:,t+tau], axis=0)
        Meanxx[tau,t] = xx
```

[ ]:
```
plt.figure(figsize=(6,5))
plt.imshow(Meanxx, cmap="viridis")
plt.colorbar()
plt.show()
```



8.7 G: $p(v)$

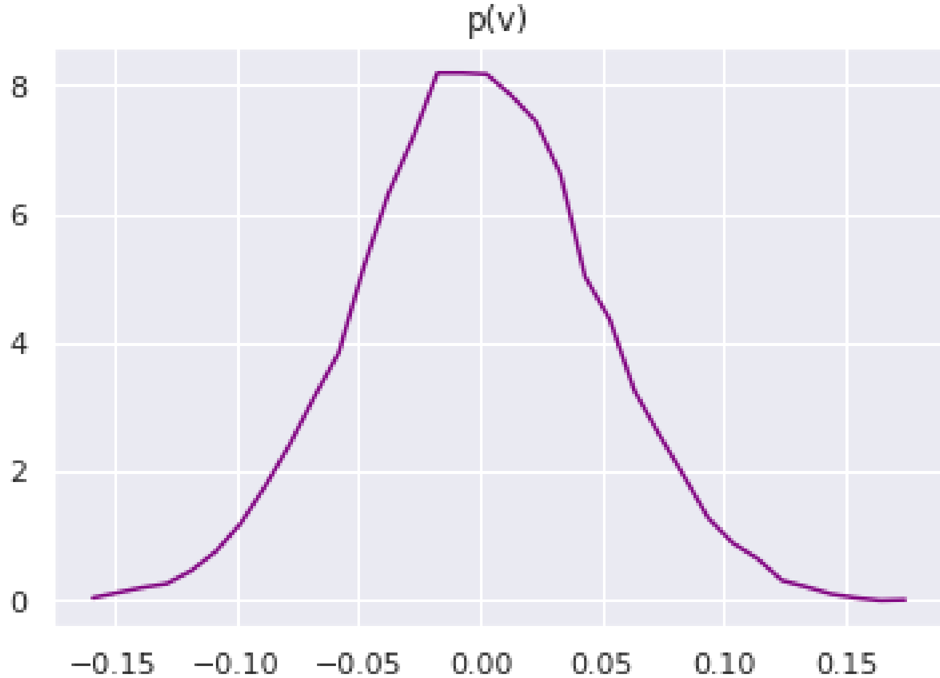
[ ]:
```
def _int(x):
    """calculate integer part of the number
    example:
        _float(-0.5) = -1
```

```
    Args:
        x (float):

    Returns:
        int:
    """
    if x>= 0:
        return int(x)
    else:
        return int(x) -1
```

[ ]:
```
def PDF(X,dx):
  """probability of data

  Args:
      X (1d_array): data
      dx (float): size of steps

  Returns:
      tuple: axis of probability and probability --> x, p(x)
  """
  n = int((X.max()-X.min())/dx) + 1

  axis = np.linspace(X.min(), X.max(), n)

  pdf = np.zeros(n)

  X -= X.min()

  for i in range(len(X)):
    k = _int(X[i]/dx)
    pdf[k] += 1

  pdf /= (np.sum(pdf)*dx)

  return axis,pdf
```

[ ]:
```
axis,pdf = PDF(meanV, 0.01)
```

[ ]:
```
plt.plot(axis,pdf, color="purple")
plt.title("p(v)")
plt.show()
```


p(v)
8
6
4
2
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

8.8 H: Compare all of above parts with theoretical predictions.

Computational $\rightsquigarrow$ $\langle v(t)\rangle = 0$, $\langle v(t)^2\rangle = 2.70$

$\langle v(t_1)\, v(t_2)\rangle = 0$ $\quad$ $\langle x(t)^2\rangle = 4t$

Analytic $\rightsquigarrow$

$$\langle v(t)\rangle_{ens} = \left\langle v_0 e^{-\xi t} + \int_0^t dt'\, e^{-\xi(t-t')}\, \eta(t')\right\rangle_{ens} = v_0 e^{-\xi t} + \int_0^t dt'\, e^{-\xi(t-t')} \cancel{\langle \eta(t')\rangle_{ens}}^{\,0}$$

$$\langle v(t)\rangle_{ens} = v_0 e^{-\xi t} \quad \rightsquigarrow \langle v(t)\rangle = 0$$

$$\langle v^2(t)\rangle_{ens} = v_0^2 e^{-2\xi t} + \int_0^t dt_1 \int_0^t dt_2\, e^{-\xi(2t-t_1-t_2)} \underbrace{\langle \eta(t_1)\eta(t_2)\rangle}_{2g\,\delta_D(t_1-t_2)}$$

$$= v_0^2 e^{-2\xi t} + \int_0^t dt_1\, 2g\, e^{-\xi(2t-2t_1)} = v_0^2 e^{-2\xi t} + \frac{2g}{2\xi}\left[1 - e^{-2\xi t}\right]$$

$$= \left[1 - e^{-2t}\right] \xrightarrow{t=1000} \langle v^2(t)\rangle_{ens} = 1$$

$$\langle v(t_1)v(t_2)\rangle = v_0^2 e^{-\xi(t_1+t_2)} + \frac{2g}{2\xi}\left[e^{-\xi|t_1-t_2|} - e^{-\xi(t_1+t_2)}\right]$$

$$\lim_{\tau = |t_1-t_2| \to \infty} \langle v(t)v(t+\tau)\rangle = 0 + \frac{g}{\xi} e^{-\xi\tau} = e^{-\tau} = 0$$

$$\langle x^2(t)\rangle = \int_0^t d\tau \int_0^t d\tau' \langle v(\tau)v(\tau')\rangle = 2t$$

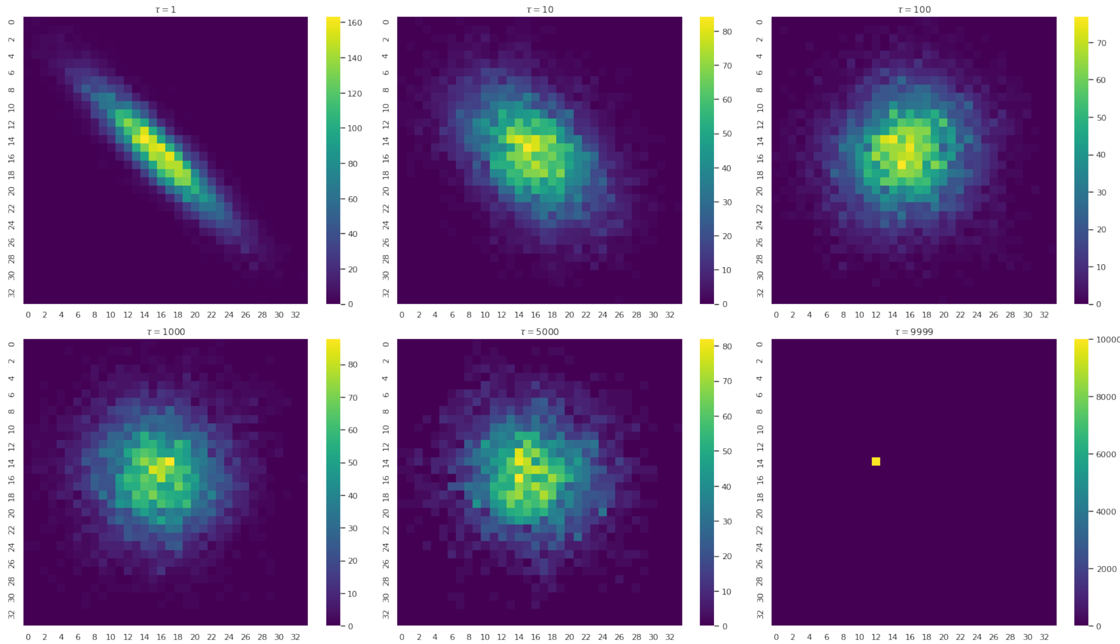
8.9 I: $p(v(t); v(t+\tau))$. What happens if $\tau \to \infty$

[ ]:
```
def p_joint(x, y, dx, dy, nx=0, ny=1, tau=0):
    """calculate joint probablity --> p(x(t+nx*tau),y(t+ny*tau))

    Args:
        x (1d_array): first data
        y (1d-array): second data
        dx (float): size of steps for x data
        dy (float): size of steps for y data
        nx (int, optional): coefficient of tau for x. Defaults to 0.
        ny (int, optional): coefficient of tau for x. Defaults to 1.
        tau (int, optional): delay time. Defaults to 0.
    Returns:
        2d_array: joint probability of x,y
    """

    numx = int((x.max()-x.min())/dx)+1
    numy = int((y.max()-y.min())/dy)+1

    pdf = np.zeros((numx, numy))
```

```
    x -= x.min()
    y -= y.min()

    for i in range(len(x)- np.max((nx,ny))*tau):
        k1 = _int(x[i+(nx*tau)]/dx)
        k2 = _int(y[i+(ny*tau)]/dy)
        pdf[k1,k2] += 1

    return pdf/(np.sum(pdf)*dx*dy)
```

[ ]:
```
p1 = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=1)
p10 = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=10)
p100 = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=100)
p1000 = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=1000)
p5000 = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=5000)
pf = p_joint(meanV, meanV, 0.01, 0.01, 0, 1, tau=9999)
```

[ ]:
```
fig, axes = plt.subplots(2, 3, figsize=(21,12))

sns.heatmap(p1, cmap="viridis", ax=axes[0,0])
axes[0,0].set_title(r"$\tau = 1$")

sns.heatmap(p10, cmap="viridis", ax=axes[0,1])
axes[0,1].set_title(r"$\tau = 10$")

sns.heatmap(p100, cmap="viridis", ax=axes[0,2])
axes[0,2].set_title(r"$\tau = 100$")

sns.heatmap(p1000, cmap="viridis", ax=axes[1,0])
axes[1,0].set_title(r"$\tau = 1000$")

sns.heatmap(p5000, cmap="viridis", ax=axes[1,1])
axes[1,1].set_title(r"$\tau = 5000$")

sns.heatmap(pf, cmap="viridis", ax=axes[1,2])
axes[1,2].set_title(r"$\tau = 9999$")

plt.tight_layout()
plt.show()
```



$\lim_{\tau \to \infty} p(v(t); v(t+\tau))$ is all zero except one point.