Homework 2, Brownian Motion

Erik Norlin, 19970807-9299

November 21, 2022

5.1: a, b

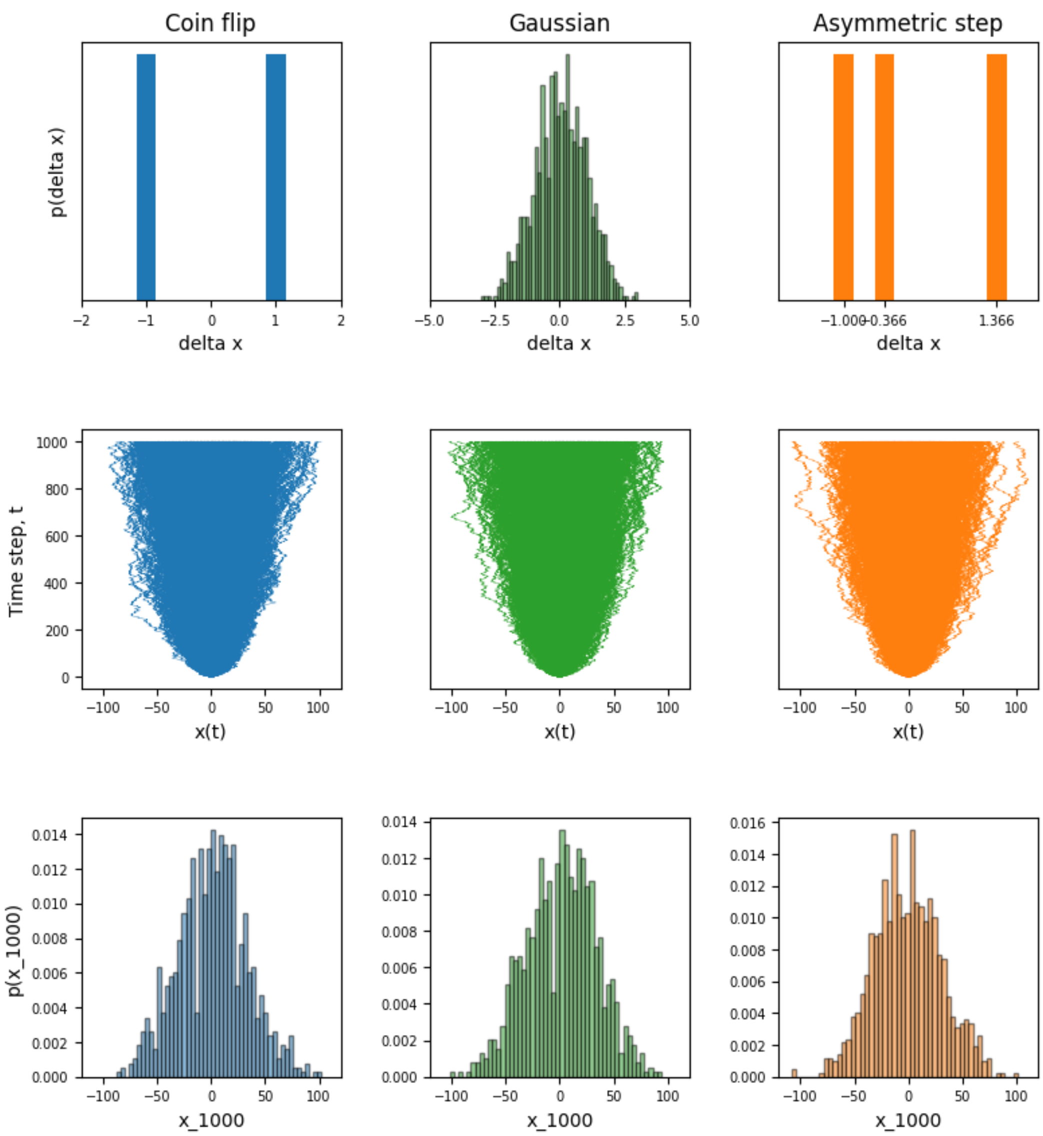

Figure 1: Sample trajectories of 1000 epochs and x_{1000}.

5.2: a, b

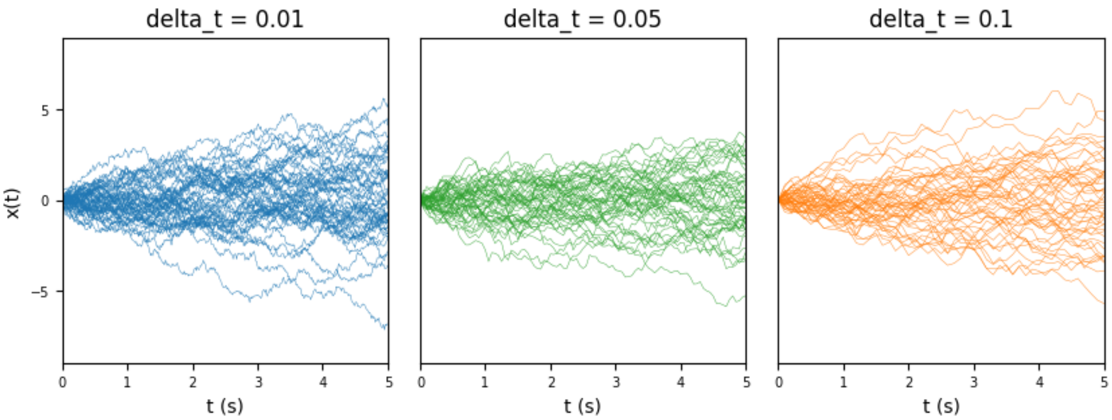

Figure 2: 50 five-second trajectories.

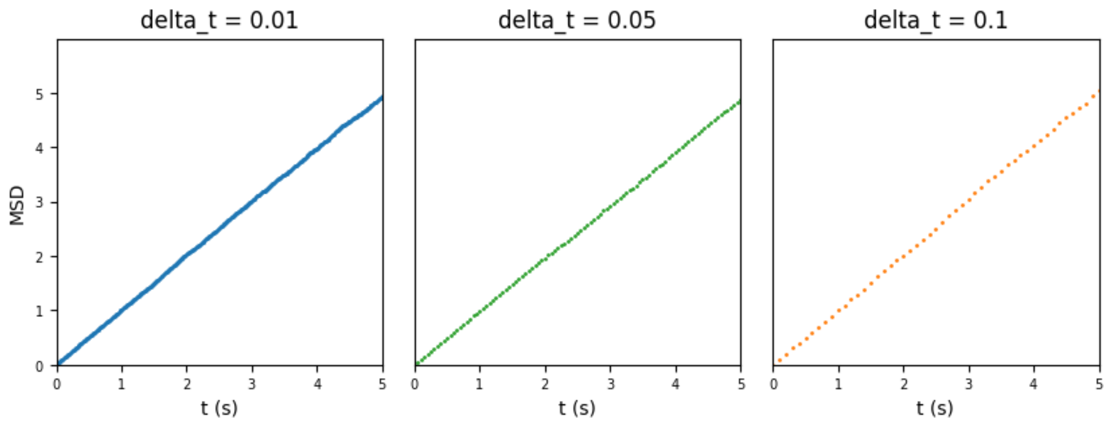

Figure 3: Ensemble averaged MSD for 10^4 trajectories.

5.3: a, b, c

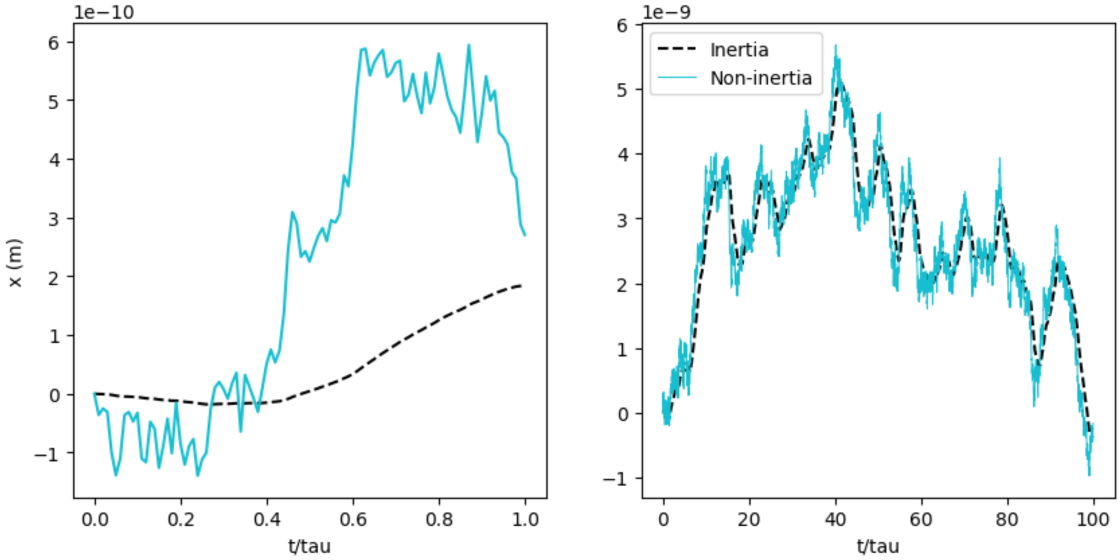

Figure 4: Simulation of Langevin equation with and without inertia (mass) for $\tau = 1$ and $\tau = 100$.

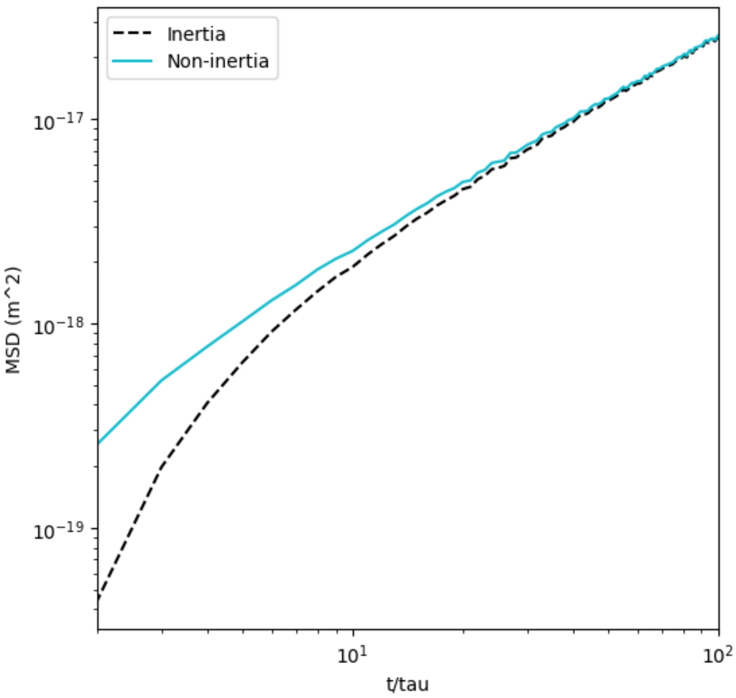

Figure 5: MSD with and without inertia for different τ-trajectories. Convergence around 20τ.

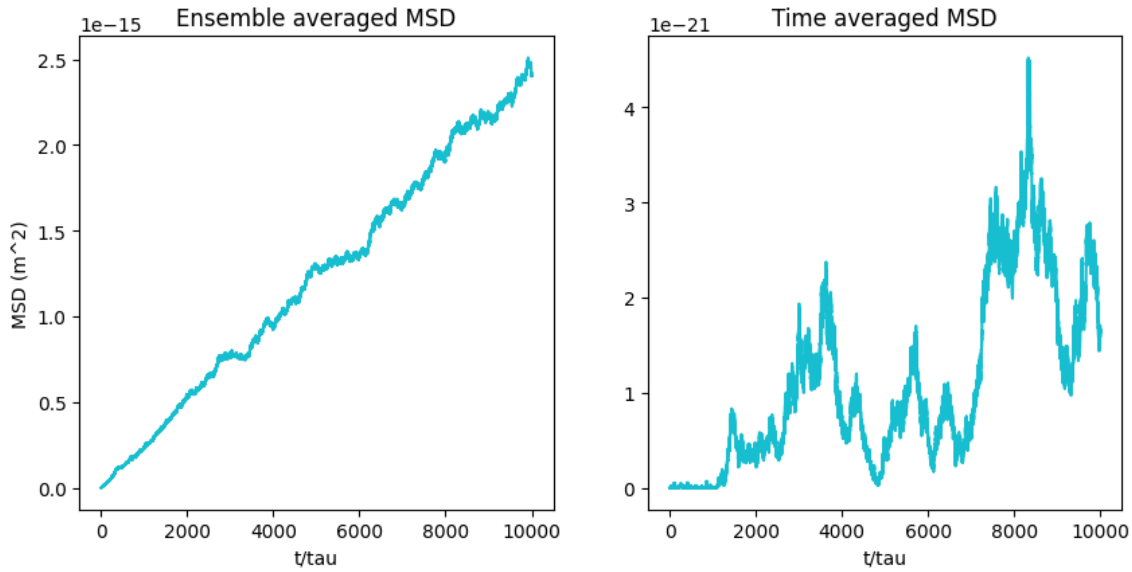

Figure 6: Ensemble MSD and time-averaged MSD over a long trajectory.

5.4: a, b, c

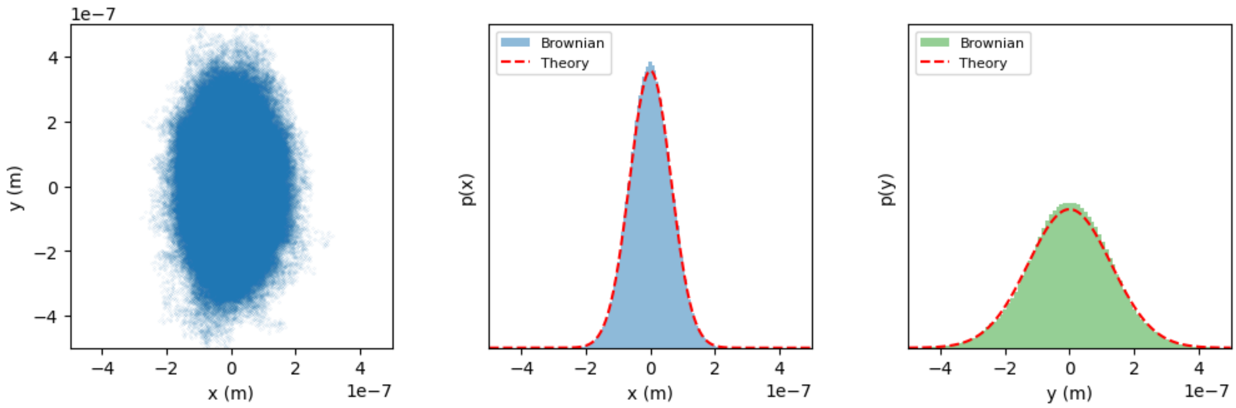

Figure 7: Simulation of brownian motion in 2D. Probability distribution for x and y and also the theory of the Boltzmann distribution for x and y.

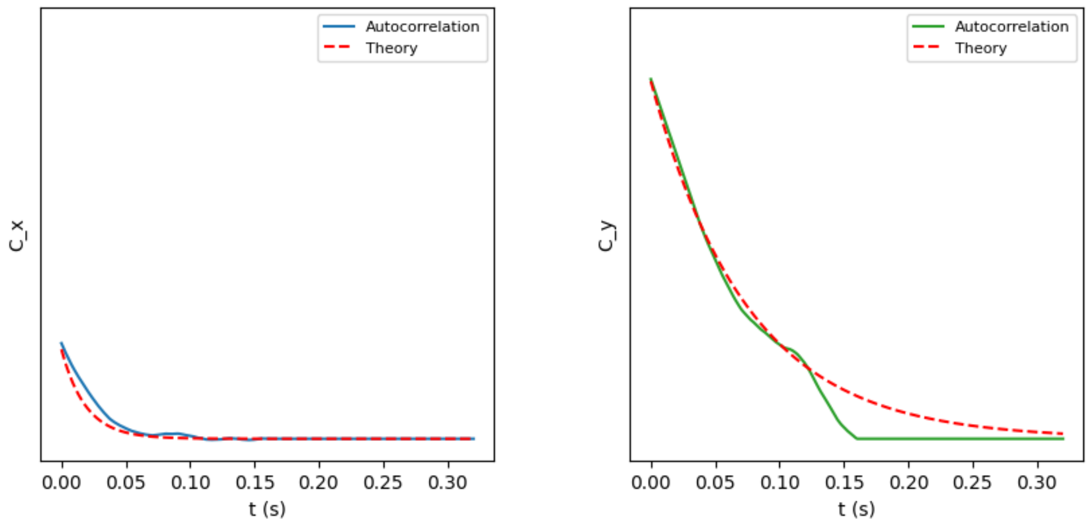

Figure 8: Positional autocorrelations $C_x(t)$ and $C_y(t)$.

```python
# Exercise 51ab

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

fig, axs = plt.subplots(3, 3, figsize=(10,10))
epochs = 1000
iterations = np.linspace(0,1000,1001)
x = np.zeros((epochs,1001))


# Coinflip
p_coinflip = 0.5
x_coinflip = x.copy()
x_coinflip[0] = 0

for epoch in range(epochs):
    for i in range(len(iterations)-1):
        rnd = np.sign(np.random.uniform() - 0.5)
        x_coinflip[epoch,i+1] = x_coinflip[epoch,i] + rnd

    axs[1,0].plot(x_coinflip[epoch,:], iterations, 'tab:blue', linewidth=0.3)

axs[0,0].bar([-1, 1], [0.5, 0.5], width=0.3, align='center', color='tab:blue')
axs[0,0].set_xlabel('delta x')
axs[0,0].set_ylabel('p(delta x)')
axs[0,0].set_box_aspect(1)
axs[0,0].set_xlim([-2,2])
axs[0,0].set_xticks((-2, -1, 0, 1, 2))
axs[0,0].set_yticks(())
axs[0,0].xaxis.set_tick_params(labelsize=7)
axs[0,0].yaxis.set_tick_params(labelsize=7)

axs[1,0].set_xlabel('x(t)')
axs[1,0].set_ylabel('Time step, t')
axs[1,0].set_box_aspect(1)
axs[1,0].set_xlim([-120,120])
axs[1,0].set_xticks((-100, -50, 0, 50, 100))
axs[1,0].set_yticks((0, 200, 400, 600, 800, 1000))
axs[1,0].xaxis.set_tick_params(labelsize=7)
axs[1,0].yaxis.set_tick_params(labelsize=7)

axs[2,0].hist(x_coinflip[:,-1], density=True, bins=50, color='tab:blue', alpha=0.5,
histtype='bar', ec='black')
axs[2,0].set_xlabel('x_1000')
axs[2,0].set_ylabel('p(x_1000)')
axs[2,0].set_box_aspect(1)
axs[2,0].set_xlim([-120,120])
axs[2,0].set_xticks((-100, -50, 0, 50, 100))
# axs[2,0].set_yticks(())
axs[2,0].xaxis.set_tick_params(labelsize=7)
axs[2,0].yaxis.set_tick_params(labelsize=7)


# Gaussian
x_gaussian = x.copy()
x_gaussian[0] = 0

```

```
for epoch in range(epochs):
    for i in range(len(iterations)-1):
        rnd = np.sign(np.random.random() - 0.5)
        x_gaussian[epoch,i+1] = x_gaussian[epoch,i] + rnd

    axs[1,1].plot(x_gaussian[epoch,:], iterations, 'tab:green', linewidth=0.3)

N = 1000
x2 = [np.random.normal() for _ in range(N)]
axs[0,1].hist(x2, bins=50, color='tab:green', alpha=0.5, histtype='bar', ec='black')
axs[0,1].set_xlabel('delta x')
axs[0,1].set_box_aspect(1)
axs[0,1].set_xlim([-5,5])
axs[0,1].set_xticks((-5, -2.5, 0, 2.5, 5))
axs[0,1].set_yticks(())
axs[0,1].xaxis.set_tick_params(labelsize=7)
axs[0,1].yaxis.set_tick_params(labelsize=7)

axs[1,1].set_xlabel('x(t)')
axs[1,1].set_box_aspect(1)
axs[1,1].set_xlim([-120,120])
axs[1,1].set_xticks((-100, -50, 0, 50, 100))
axs[1,1].set_yticks(())
axs[1,1].xaxis.set_tick_params(labelsize=7)
axs[1,1].yaxis.set_tick_params(labelsize=7)

axs[2,1].hist(x_gaussian[:,-1], density=True, bins=50, color='tab:green', alpha=0.5,
histtype='bar', ec='black')
axs[2,1].set_xlabel('x_1000')
axs[2,1].set_box_aspect(1)
axs[2,1].set_xlim([-120,120])
axs[2,1].set_xticks((-100, -50, 0, 50, 100))
# axs[2,1].set_yticks(())
axs[2,1].xaxis.set_tick_params(labelsize=7)
axs[2,1].yaxis.set_tick_params(labelsize=7)


# Step
x_step = x.copy()
x_step[0] = 0

for epoch in range(epochs):
    for i in range(len(iterations)-1):
        rnd = np.random.random()
        if rnd < 1/3:
            x_step[epoch,i+1] = x_step[epoch,i] + (-1)
        elif rnd < 2/3:
            x_step[epoch,i+1] = x_step[epoch,i] + (1 - np.sqrt(3))/2
        elif rnd < 3/3:
            x_step[epoch,i+1] = x_step[epoch,i] + (1 + np.sqrt(3))/2

    axs[1,2].plot(x_step[epoch,:], iterations, 'tab:orange', linewidth=0.3)

a = (1 - np.sqrt(3))/2
b = (1 + np.sqrt(3))/2
axs[0,2].bar([-1, a, b], [1/3, 1/3, 1/3], width=0.3, align='center',
color='tab:orange')
axs[0,2].set_xlabel('delta x')
axs[0,2].set_box_aspect(1)
axs[0,2].set_xlim([-2,2])
```

```
axs[0,2].set_xticks((-1, a, b))
axs[0,2].set_yticks(())
axs[0,2].xaxis.set_tick_params(labelsize=7)
axs[0,2].yaxis.set_tick_params(labelsize=7)

axs[1,2].set_xlabel('x(t)')
axs[1,2].set_box_aspect(1)
axs[1,2].set_xlim([-120,120])
axs[1,2].set_xticks((-100, -50, 0, 50, 100))
axs[1,2].set_yticks(())
axs[1,2].xaxis.set_tick_params(labelsize=7)
axs[1,2].yaxis.set_tick_params(labelsize=7)

axs[2,2].hist(x_step[:,-1], density=True, bins=50, color='tab:orange', alpha=0.5,
histtype='bar', ec='black')
axs[2,2].set_xlabel('x_1000')
axs[2,2].set_box_aspect(1)
axs[2,2].set_xlim([-120,120])
axs[2,2].set_xticks((-100, -50, 0, 50, 100))
# axs[2,2].set_yticks(())
axs[2,2].xaxis.set_tick_params(labelsize=7)
axs[2,2].yaxis.set_tick_params(labelsize=7)


axs[0,0].set_title('Coin flip')
axs[0,1].set_title('Gaussian')
axs[0,2].set_title('Asymmetric step')

matplotlib.rc('xtick', labelsize=5)
matplotlib.rc('ytick', labelsize=5)

plt.subplots_adjust(wspace=0.01, hspace=0.5)
plt.savefig('exercise_51ab.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 52a

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

fig, axs = plt.subplots(1, 3, figsize=(10,10))
epochs = 50

delta_t_array = np.array([0.01, 0.05, 0.1])
color_array = np.array(['tab:blue', 'tab:green', 'tab:orange'])
time_trajectory = 5

for delta_t_i in range(len(delta_t_array)):

    delta_t = delta_t_array[delta_t_i]
    no_iterations = int(time_trajectory / delta_t)
    iterations = np.linspace(0, no_iterations, no_iterations+1)
    time_scale = no_iterations / time_trajectory

    x_t = np.zeros((epochs, no_iterations+1))
    x_t[:,0] = 0

    for epoch in range(epochs):
        for i in range(len(iterations)-1):
            rnd = np.random.normal()
            x_t[epoch,i+1] = x_t[epoch,i] + (rnd * np.sqrt(delta_t))

        axs[delta_t_i].plot(iterations/time_scale, x_t[epoch,:], color=color_array[delta_t_i], linewidth=0.3)

    axs[delta_t_i].set_xlabel('t (s)')
    axs[delta_t_i].set_box_aspect(1)
    axs[delta_t_i].set_ylim([-9,9])
    axs[delta_t_i].set_xlim([0,5])
    axs[delta_t_i].set_xticks((0, 1, 2, 3, 4, 5))
    axs[delta_t_i].set_yticks(())
    axs[delta_t_i].xaxis.set_tick_params(labelsize=7)
    axs[delta_t_i].yaxis.set_tick_params(labelsize=7)

axs[0].set_yticks((-5, 0, 5))
axs[0].set_ylabel('x(t)')
axs[0].set_title('delta_t = 0.01')
axs[1].set_title('delta_t = 0.05')
axs[2].set_title('delta_t = 0.1')

matplotlib.rc('xtick', labelsize=5)
matplotlib.rc('ytick', labelsize=5)

plt.subplots_adjust(wspace=0.1, hspace=0.5)
plt.savefig('exercise_52a.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 52b

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

fig, axs = plt.subplots(1, 3, figsize=(10,10))
epochs = 10**4

delta_t_array = np.array([0.01, 0.05, 0.1])
color_array = np.array(['tab:blue', 'tab:green', 'tab:orange'])
time_trajectory = 10

for delta_t_i in range(len(delta_t_array)):

    delta_t = delta_t_array[delta_t_i]
    no_iterations = int(time_trajectory / delta_t)
    iterations = np.linspace(0, no_iterations, no_iterations+1)
    time_scale = no_iterations / time_trajectory

    x_t = np.zeros((epochs, no_iterations+1))
    x_t[:,0] = 0

    MSD = np.zeros((epochs, len(iterations)))
    MSD[:,0] = 0

    for epoch in range(epochs):
        for i in range(len(iterations)-1):
            rnd = np.random.normal()
            x_t[epoch,i+1] = x_t[epoch,i] + (rnd * np.sqrt(delta_t))
            MSD[epoch,i+1] = np.power(x_t[epoch,i+1], 2)

    MSD_average = np.zeros(len(MSD[0,:]))
    for i in range(len(MSD_average)):
        MSD_average[i] = np.sum(MSD[:,i]) / len(MSD[:,i])

    axs[delta_t_i].plot(iterations/time_scale, MSD_average, '.', markersize=2,
color=color_array[delta_t_i] )
    axs[delta_t_i].set_xlabel('t (s)')
    axs[delta_t_i].set_box_aspect(1)
    axs[delta_t_i].set_ylim([0,6])
    axs[delta_t_i].set_xlim([0,5])
    axs[delta_t_i].set_xticks((0, 1, 2, 3, 4, 5))
    axs[delta_t_i].set_yticks(())
    axs[delta_t_i].xaxis.set_tick_params(labelsize=7)
    axs[delta_t_i].yaxis.set_tick_params(labelsize=7)

axs[0].set_yticks((0, 1, 2, 3, 4, 5))
axs[0].set_ylabel('MSD')
axs[0].set_title('delta_t = 0.01')
axs[1].set_title('delta_t = 0.05')
axs[2].set_title('delta_t = 0.1')

matplotlib.rc('xtick', labelsize=5)
matplotlib.rc('ytick', labelsize=5)

plt.subplots_adjust(wspace=0.1, hspace=0.5)
plt.savefig('exercise_52b.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 53a

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

n = 0.001
R = 1/10**6
pi = np.pi
gamma = 6*pi*R*n
T = 300
k = 1.380649/10**23
m = 1.11/10**14
tau = m/gamma
no_iterations = 100
dt = tau*0.01

# 1 tau long trajectory
timesteps = int(tau/dt)
iterations = np.linspace(0, 1, timesteps+1)
x = np.zeros(timesteps+1)
x_weight = x.copy()
x_weightless = x.copy()

for i in range(timesteps):
    w = np.random.normal()
    x_weight[i+1] = (x_weight[i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) - (x_weight[i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
    x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))

fig, axs = plt.subplots(1,2, figsize=(10,10))

axs[0].plot(iterations, x_weight, '--', color='black', label='Inertia')
axs[0].plot(iterations, x_weightless, color='tab:cyan', label='Non-inertia')
axs[0].set_xlabel('t/tau')
axs[0].set_ylabel('x (m)')
axs[0].set_box_aspect(1)


# 100 tau long trajectory
timesteps = int(tau/dt)*100
iterations = np.linspace(0, 100, timesteps+1)
x = np.zeros(timesteps+1)
x_weight = x.copy()
x_weightless = x.copy()

for i in range(timesteps):
    w = np.random.normal()
    x_weight[i+1] = (x_weight[i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) - (x_weight[i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
    x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))

axs[1].plot(iterations, x_weight, '--', color='black', label='Inertia')
axs[1].plot(iterations, x_weightless, color='tab:cyan', linewidth=0.7, label='Non-inertia')
axs[1].set_xlabel('t/tau')
```

```python
axs[1].set_box_aspect(1)

plt.legend(loc="upper left")
plt.savefig('exercise_53a.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 53b

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

n = 0.001
R = 1/10**6
pi = np.pi
gamma = 6*pi*R*n
T = 300
k = 1.380649/10**23
m = 1.11/10**14
tau = m/gamma
no_iterations = 100
time_step = 0.01
dt = tau*time_step

epochs = 10000
tau_trajectory = np.linspace(1,no_iterations,no_iterations)
x_weight_MSD = np.zeros(len(tau_trajectory))
x_weightless_MSD = np.zeros(len(tau_trajectory))
fig, ax = plt.subplots(figsize=(6,6))

for i_tau in range(len(tau_trajectory)):

    timesteps = int(tau/dt)*i_tau
    iterations = np.linspace(0, timesteps*time_step, timesteps+1)

    x = np.zeros((epochs,timesteps+1))
    x_weight = x.copy()
    x_weightless = x.copy()

    for epoch in range(epochs):

        for i in range(timesteps):
            w = np.random.normal()
            x_weight[epoch, i+1] = (x_weight[epoch, i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) - (x_weight[epoch, i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
            x_weightless[epoch, i+1] = x_weightless[epoch, i] + (w * np.sqrt(2*k*T*dt/gamma))

    x_weight_MSD[i_tau] = np.sum(x_weight[:,-1]**2) / epochs
    x_weightless_MSD[i_tau] = np.sum(x_weightless[:,-1]**2) / epochs

    print(i_tau)

ax.plot(tau_trajectory, x_weight_MSD, '--', color='black', label='Inertia')
ax.plot(tau_trajectory, x_weightless_MSD, '-', color='tab:cyan', label='Non-inertia')
ax.set_xlabel('t/tau')
ax.set_ylabel('MSD (m^2)')
ax.set_box_aspect(1)
ax.set_yscale('log')
ax.set_xscale('log')
ax.set_xlim([2,100])
# 'linear', 'log', 'symlog', 'asinh', 'logit', 'function', 'functionlog'
plt.legend(loc="upper left")
```

```python
plt.savefig('exercise_53bv2.png', bbox_inches='tight')
plt.show()
```

```
# Exercise 53c

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

n = 0.001
R = 1/10**6
pi = np.pi
gamma = 6*pi*R*n
T = 300
k = 1.380649/10**23
m = 1.11/10**14
tau = m/gamma
time_step = 0.01
dt = tau*time_step
tau_trajectory = 10000
timesteps = int(tau/dt)*tau_trajectory
iterations = np.linspace(0, timesteps*time_step, timesteps+1)
epochs = 100

# Ensemble average MSD
x = np.zeros((epochs,timesteps+1))
# x_weight = x.copy()
x_weightless = x.copy()
x_weight_ensemble_MSD = np.zeros(timesteps+1)
x_weightless_ensemble_MSD = np.zeros(timesteps+1)

for epoch in range(epochs):

    for i in range(timesteps):
        w = np.random.normal()
        # x_weight[epoch,i+1] = (x_weight[epoch,i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) - (x_weight[epoch,i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
        x_weightless[epoch,i+1] = x_weightless[epoch,i] + (w * np.sqrt(2*k*T*dt/gamma))

    print(epoch)

for i in range(timesteps):
    # x_weight_ensemble_MSD[i+1] = np.sum(x_weight[:,i]**2) / epochs
    x_weightless_ensemble_MSD[i+1] = np.sum(x_weightless[:,i]**2) / epochs

fig, axs = plt.subplots(1,2,figsize=(10,10))

# axs[0].plot(iterations, x_weight_ensemble_MSD, '--', color='black')
axs[0].plot(iterations, x_weightless_ensemble_MSD, '-', color='tab:cyan')
axs[0].set_xlabel('t/tau')
axs[0].set_ylabel('MSD (m^2)')
axs[0].set_title('Ensemble averaged MSD')
axs[0].set_box_aspect(1)

# Time average MSD
x = np.zeros((timesteps+1))
x_weight = x.copy()
x_weightless = x.copy()
x_weight_time_MSD = np.zeros(timesteps+1)
```

```python
x_weightless_time_MSD = np.zeros(timesteps+1)

for i in range(timesteps):
    w = np.random.normal()
    # x_weight[i+1] = (x_weight[i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) - (x_weight[i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
    x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))

    # x_weight_time_MSD[i+1] = x_weight[i]**2 / timesteps
    x_weightless_time_MSD[i+1]= x_weightless[i]**2 / timesteps

    if i % 1000 == 0:
        print(i)

# axs[1].plot(iterations, x_weight_time_MSD, '--', color='black', label='Inertia')
axs[1].plot(iterations, x_weightless_time_MSD, '-', color='tab:cyan', label='Non-inertia')
axs[1].set_xlabel('t/tau')
axs[1].set_box_aspect(1)
axs[1].set_title('Time averaged MSD')

# plt.legend(loc="upper left")
plt.savefig('exercise_53c.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 54ab

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

n = 0.001
R = 1/10**6
gamma = 6*np.pi*n*R
k = 1.380649/10**23
T = 300
kx = (1/10**12) / (1/10**6)
ky = (0.25/10**12) / (1/10**6)
tau = gamma / kx # Arbitary choose kx
time_step = 0.01
dt = tau*time_step
tau_trajectory = 10000
timesteps = int(tau/dt)*tau_trajectory
iterations = np.linspace(0, timesteps*time_step, timesteps+1)
x = np.zeros(timesteps+1)
y = np.zeros(timesteps+1)

for i in range(timesteps-1):
    w = np.random.normal()
    x[i+1] = x[i] - (dt*x[i]*kx/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
    w = np.random.normal()
    y[i+1] = y[i] - (dt*y[i]*ky/gamma) + (w*np.sqrt(2*k*T*dt/gamma))

xmax = 5/10**7
ymax = 7*10**6
x_sym = np.linspace(-xmax, xmax, timesteps)
y_sym = x_sym
px = 6*10**6*np.exp(-(0.5*kx*x_sym**2)/(k*T))
py = 3*10**6*np.exp(-(0.5*ky*y_sym**2)/(k*T))

fig, axs = plt.subplots(1,3,figsize=(12,12))

axs[0].plot(x, y, '.', color='tab:blue', markersize=0.1)
axs[0].set_xlabel('x (m)')
axs[0].set_ylabel('y (m)')
axs[0].set_xlim([-xmax, xmax])
axs[0].set_ylim([-xmax, xmax])
axs[0].set_box_aspect(1)

weights_x = 30*np.ones_like(x)/len(x)
weights_y = 30*np.ones_like(y)/len(y)

axs[1].hist(x, density=True, bins=100, color='tab:blue', alpha=0.5, label='Brownian')
axs[1].plot(x_sym, px, '--', color='red', label='Theory')
axs[1].set_xlabel('x (m)')
axs[1].set_ylabel('p(x)')
axs[1].set_box_aspect(1)
axs[1].set_xlim([-xmax, xmax])
axs[1].legend(loc="upper left",prop={'size': 8})
axs[1].set_ylim([0, ymax])
axs[1].set_yticks(())

```

```
axs[2].hist(y, density=True, bins=100, color='tab:green', alpha=0.5,
label='Brownian')
axs[2].plot(y_sym, py, '--', color='red', label='Theory')
axs[2].set_xlabel('y (m)')
axs[2].set_ylabel('p(y)')
axs[2].set_box_aspect(1)
axs[2].set_xlim([-xmax, xmax])
axs[2].legend(loc="upper left",prop={'size': 8})
axs[2].set_ylim([0, ymax])
axs[2].set_yticks(())

plt.subplots_adjust(wspace=0.3, hspace=0.5)
plt.savefig('exercise_54ab.png', bbox_inches='tight')
plt.show()
```

```python
# Exercise 54c

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

n = 0.001
R = 1/10**6
gamma = 6*np.pi*n*R
k = 1.380649/10**23
T = 300
kx = (1/10**12) / (1/10**6)
ky = (0.25/10**12) / (1/10**6)
tau = gamma / kx # Arbitary choose kx
time_step = 0.001
dt = tau*time_step
tau_trajectory = 10
timesteps = int(tau/dt)*tau_trajectory
iterations = np.linspace(0, timesteps*time_step, timesteps+1)
x = np.zeros(timesteps+1)
y = np.zeros(timesteps+1)
epochs = 10
cx = np.zeros((epochs,timesteps+1))
cy = np.zeros((epochs,timesteps+1))
cx_average = np.zeros(timesteps+1)
cy_average = np.zeros(timesteps+1)

for epoch in range(epochs):
    for i in range(timesteps-1):
        w = np.random.normal()
        x[i+1] = x[i] - (dt*x[i]*kx/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
        w = np.random.normal()
        y[i+1] = y[i] - (dt*y[i]*ky/gamma) + (w*np.sqrt(2*k*T*dt/gamma))

    print(timesteps)
    for i in range(timesteps):
        if i % 100 == 0:
            print(i)
        for j in range(timesteps-i):
            cx[epoch,j] = cx[epoch,j] + x[i]*x[i+j]
            cy[epoch,j] = cy[epoch,j] + y[i]*y[i+j]
        cx[epoch,i] = cx[epoch,i]/(timesteps-i)
        cy[epoch,i] = cy[epoch,i]/(timesteps-i)
    cx[epoch,:] = cx[epoch,:]/timesteps
    cy[epoch,:] = cy[epoch,:]/timesteps

    print(epoch)

for i in range(timesteps):
    cx_average[i] = np.sum(cx[:,i]) / epochs
    cy_average[i] = np.sum(cy[:,i]) / epochs

t = np.linspace(0, 17*tau, timesteps+1)
cx_theory = (k*T/kx)*np.exp(-(kx*t/gamma))
cy_theory = (k*T/ky)*np.exp(-(ky*t/gamma))

fig, ax = plt.subplots(1,2,figsize=(10,10))

```

```python
ax[0].plot(t, cx_average, '-', color='tab:blue', label='Autocorrelation')
ax[0].plot(t, cx_theory, '--', color='red', label='Theory')
ax[0].set_xlabel('t (s)')
ax[0].set_ylabel('C_x')
ymax = 2/10**14
ymin = -1/10**15
ax[0].set_ylim([ymin,ymax])
ax[0].legend(loc="upper right",prop={'size': 8})
ax[0].set_box_aspect(1)
ax[0].set_yticks(())

ax[1].plot(t, cy_average, '-', color='tab:green', label='Autocorrelation')
ax[1].plot(t, cy_theory, '--', color='red', label='Theory')
ax[1].set_xlabel('t (s)')
ax[1].set_ylabel('C_y')
ax[1].set_yticks(())
ax[1].set_ylim([ymin,ymax])
ax[1].legend(loc="upper right",prop={'size': 8})
ax[1].set_box_aspect(1)

plt.subplots_adjust(wspace=0.3, hspace=0.5)
plt.savefig('exercise_54c.png', bbox_inches='tight')
plt.show()
```