

Homework 2, Brownian Motion

Erik Norlin, 19970807-9299

November 21, 2022

5.1: a, b

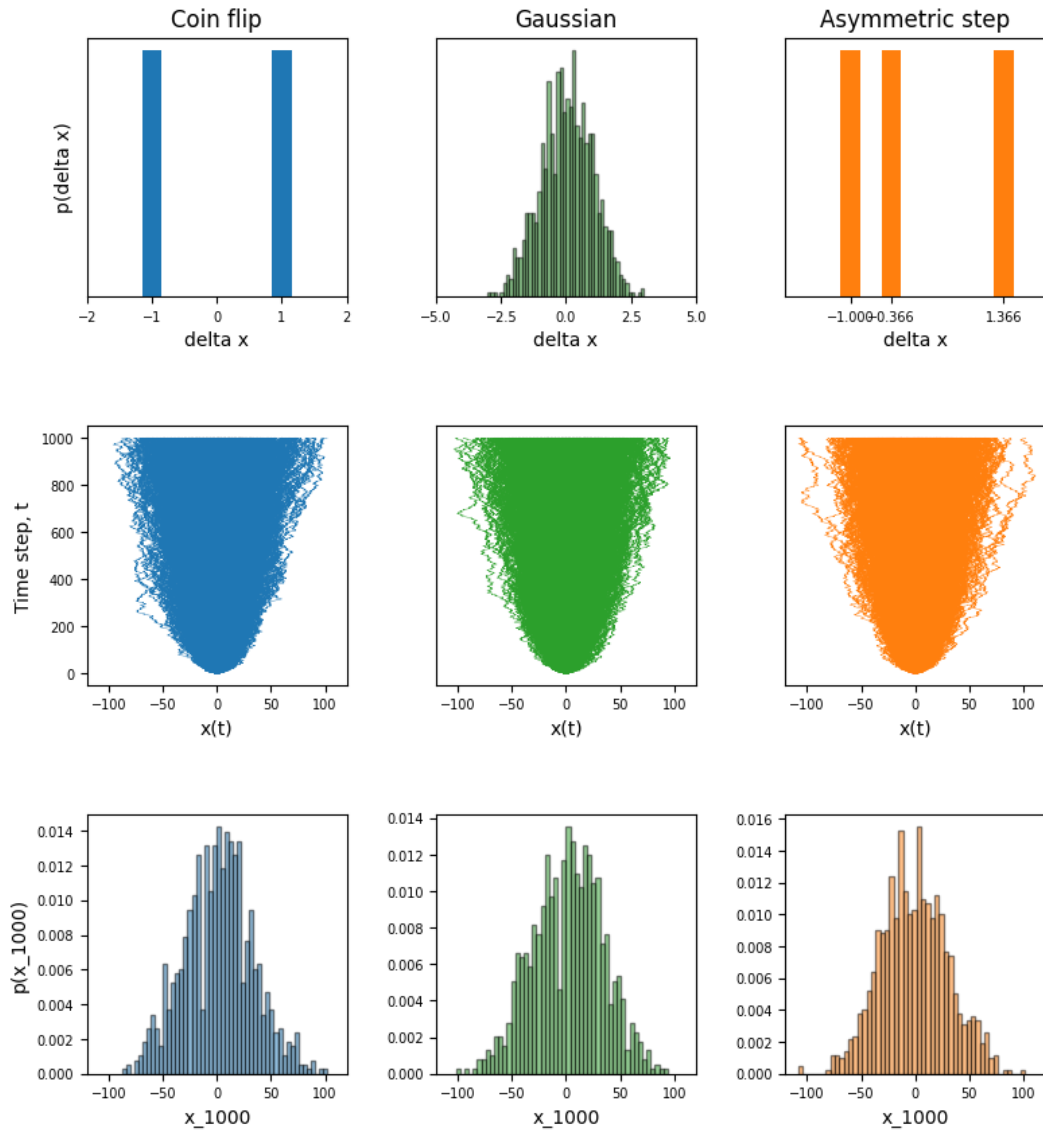


Figure 1: Sample trajectories of 1000 epochs and x_{1000} .

5.2: a, b

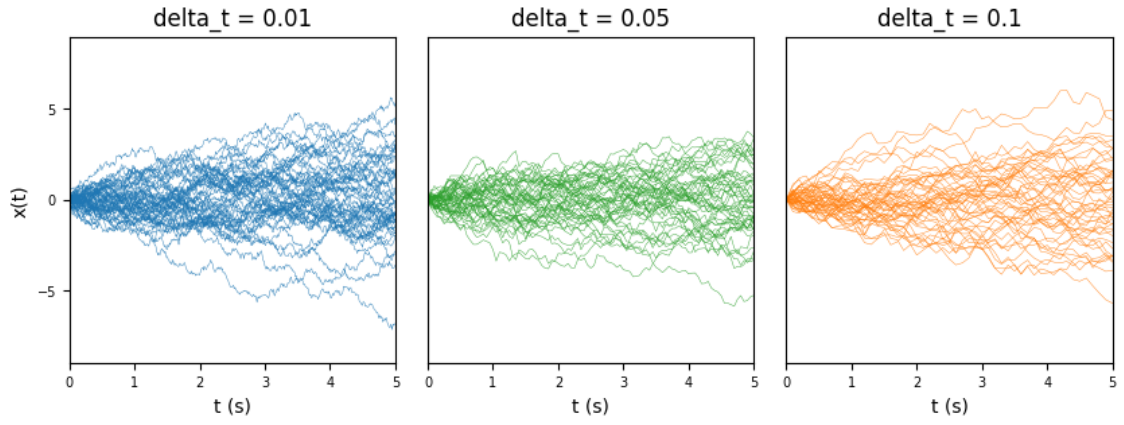


Figure 2: 50 five-second trajectories.

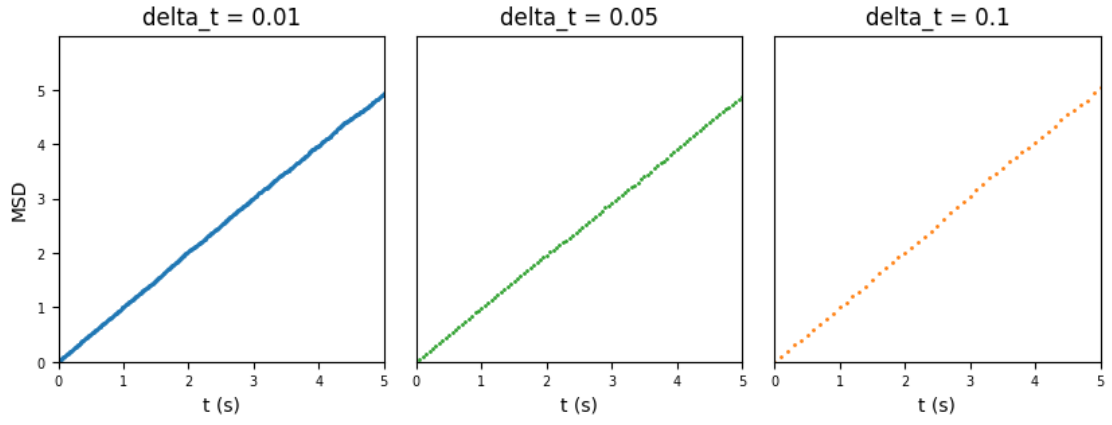


Figure 3: Ensemble averaged MSD for 10^4 trajectories.

5.3: a, b, c

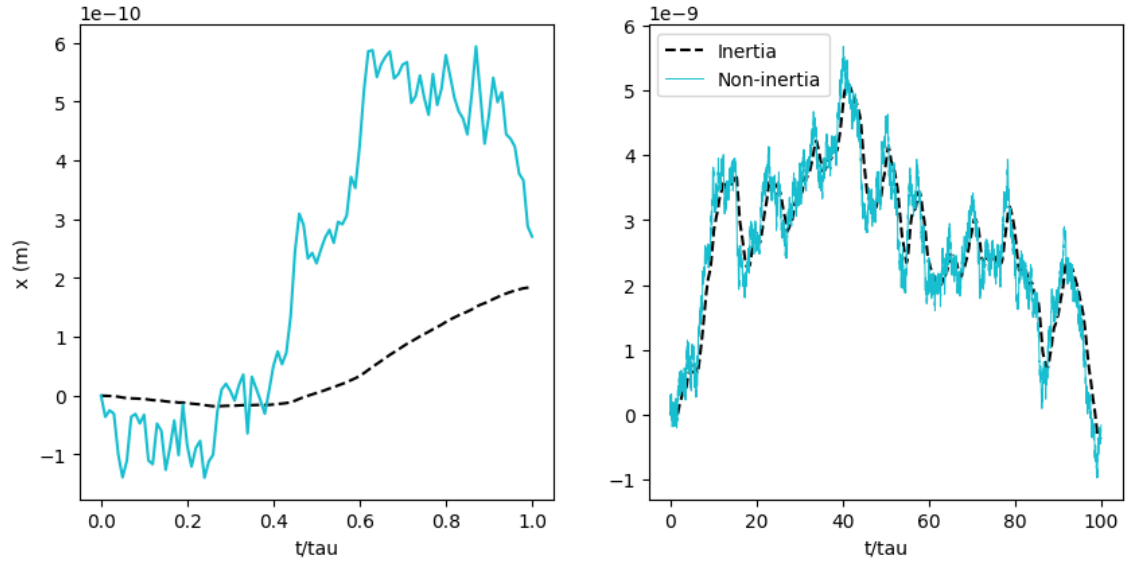


Figure 4: Simulation of Langevin equation with and without inertia (mass) for $\tau = 1$ and $\tau = 100$.

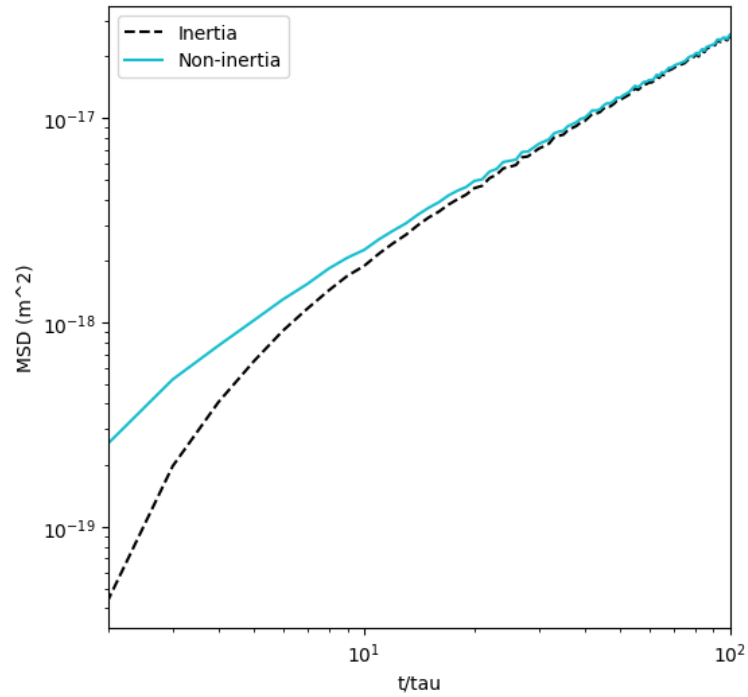


Figure 5: MSD with and without inertia for different τ -trajectories. Convergence around 20τ .

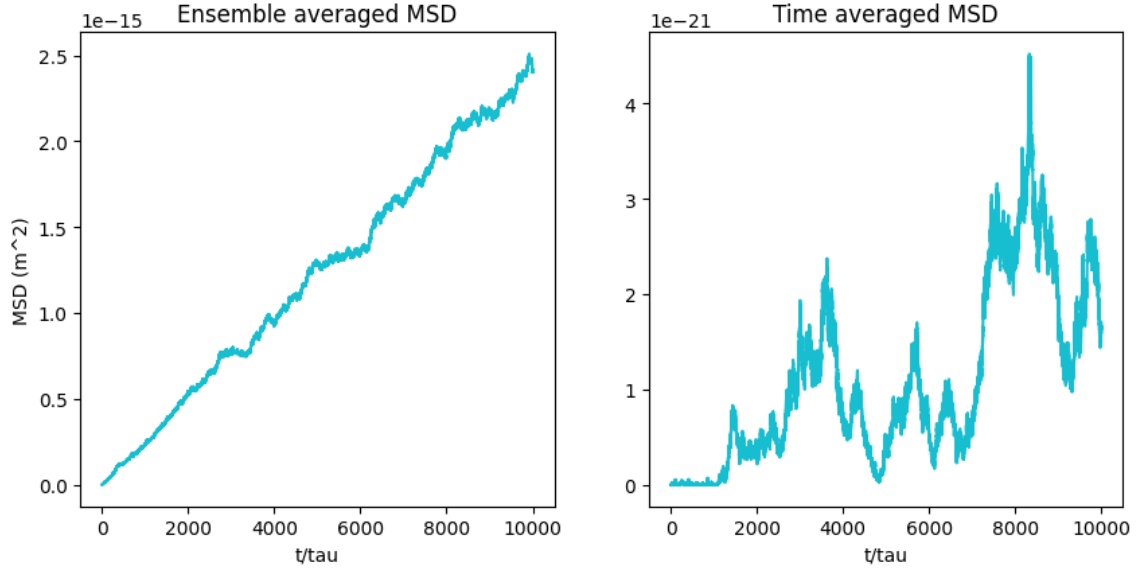


Figure 6: Ensemble MSD and time-averaged MSD over a long trajectory.

5.4: a, b, c

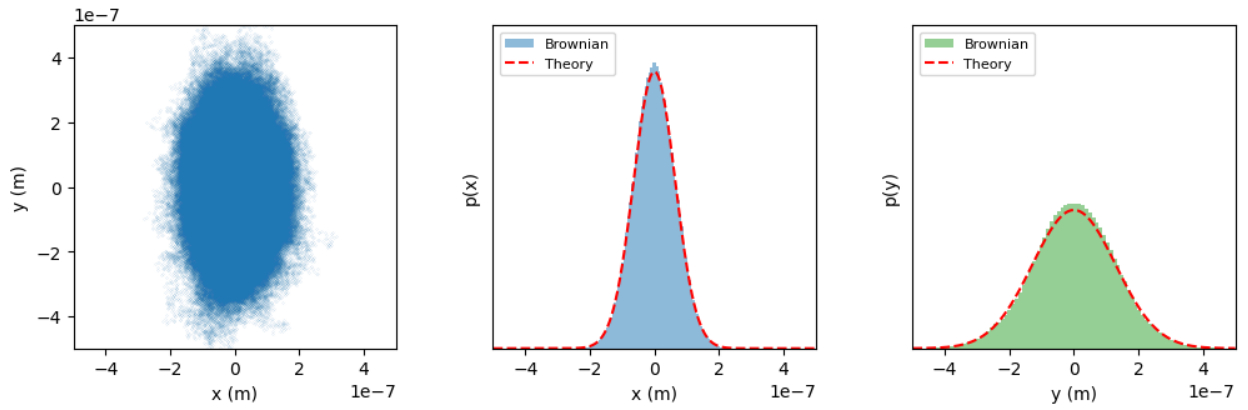


Figure 7: Simulation of brownian motion in 2D. Probability distribution for x and y and also the theory of the Boltzmann distribution for x and y.

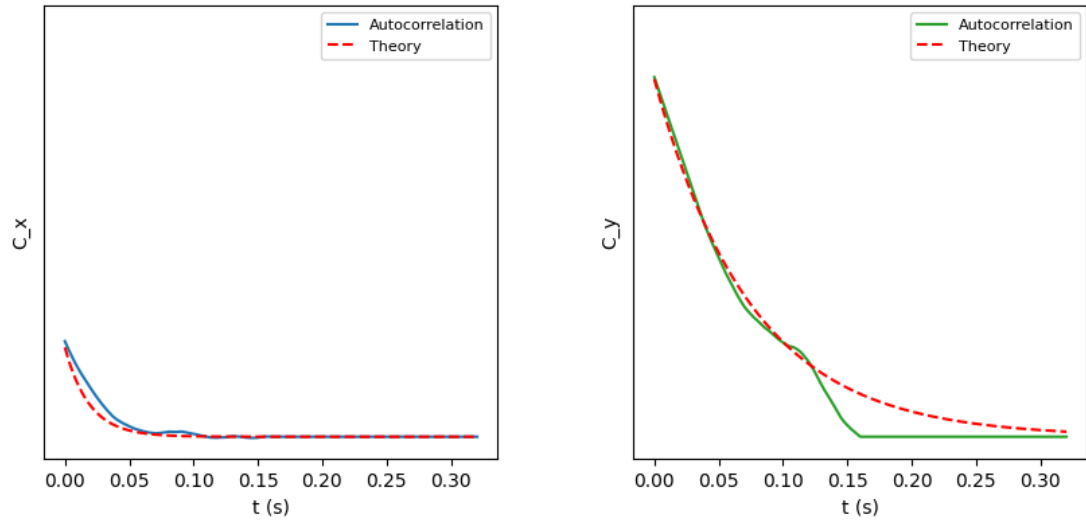


Figure 8: Positional autocorrelations $C_x(t)$ and $C_y(t)$.

```

1 # Exercise 51ab
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6
7 fig, axs = plt.subplots(3, 3, figsize=(10,10))
8 epochs = 1000
9 iterations = np.linspace(0,1000,1001)
10 x = np.zeros((epochs,1001))
11
12
13 # Coinflip
14 p_coinflip = 0.5
15 x_coinflip = x.copy()
16 x_coinflip[0] = 0
17
18 for epoch in range(epochs):
19     for i in range(len(iterations)-1):
20         rnd = np.sign(np.random.uniform() - 0.5)
21         x_coinflip[epoch,i+1] = x_coinflip[epoch,i] + rnd
22
23     axs[1,0].plot(x_coinflip[epoch,:], iterations, 'tab:blue', linewidth=0.3)
24
25 axs[0,0].bar([-1, 1], [0.5, 0.5], width=0.3, align='center', color='tab:blue')
26 axs[0,0].set_xlabel('delta x')
27 axs[0,0].set_ylabel('p(delta x)')
28 axs[0,0].set_box_aspect(1)
29 axs[0,0].set_xlim([-2,2])
30 axs[0,0].set_xticks((-2, -1, 0, 1, 2))
31 axs[0,0].set_yticks(())
32 axs[0,0].xaxis.set_tick_params(labelsize=7)
33 axs[0,0].yaxis.set_tick_params(labelsize=7)
34
35 axs[1,0].set_xlabel('x(t)')
36 axs[1,0].set_ylabel('Time step, t')
37 axs[1,0].set_box_aspect(1)
38 axs[1,0].set_xlim([-120,120])
39 axs[1,0].set_xticks((-100, -50, 0, 50, 100))
40 axs[1,0].set_yticks((0, 200, 400, 600, 800, 1000))
41 axs[1,0].xaxis.set_tick_params(labelsize=7)
42 axs[1,0].yaxis.set_tick_params(labelsize=7)
43
44 axs[2,0].hist(x_coinflip[:, -1], density=True, bins=50, color='tab:blue', alpha=0.5,
45             histtype='bar', ec='black')
46 axs[2,0].set_xlabel('x_1000')
47 axs[2,0].set_ylabel('p(x_1000)')
48 axs[2,0].set_box_aspect(1)
49 axs[2,0].set_xlim([-120,120])
50 # axs[2,0].set_yticks(())
51 axs[2,0].xaxis.set_tick_params(labelsize=7)
52 axs[2,0].yaxis.set_tick_params(labelsize=7)
53
54
55 # Gaussian
56 x_gaussian = x.copy()
57 x_gaussian[0] = 0
58

```

```

59 for epoch in range(epochs):
60     for i in range(len(iterations)-1):
61         rnd = np.sign(np.random.random() - 0.5)
62         x_gaussian[epoch,i+1] = x_gaussian[epoch,i] + rnd
63
64     axs[1,1].plot(x_gaussian[epoch,:], iterations, 'tab:green', linewidth=0.3)
65
66 N = 1000
67 x2 = [np.random.normal() for _ in range(N)]
68 axs[0,1].hist(x2, bins=50, color='tab:green', alpha=0.5, histtype='bar', ec='black')
69 axs[0,1].set_xlabel('delta x')
70 axs[0,1].set_box_aspect(1)
71 axs[0,1].set_xlim([-5,5])
72 axs[0,1].set_xticks((-5, -2.5, 0, 2.5, 5))
73 axs[0,1].set_yticks(())
74 axs[0,1].xaxis.set_tick_params(labelsize=7)
75 axs[0,1].yaxis.set_tick_params(labelsize=7)
76
77 axs[1,1].set_xlabel('x(t)')
78 axs[1,1].set_box_aspect(1)
79 axs[1,1].set_xlim([-120,120])
80 axs[1,1].set_xticks((-100, -50, 0, 50, 100))
81 axs[1,1].set_yticks(())
82 axs[1,1].xaxis.set_tick_params(labelsize=7)
83 axs[1,1].yaxis.set_tick_params(labelsize=7)
84
85 axs[2,1].hist(x_gaussian[:, -1], density=True, bins=50, color='tab:green', alpha=0.5,
86             histtype='bar', ec='black')
87 axs[2,1].set_xlabel('x_1000')
88 axs[2,1].set_box_aspect(1)
89 axs[2,1].set_xlim([-120,120])
90 axs[2,1].set_xticks((-100, -50, 0, 50, 100))
91 # axs[2,1].set_yticks(())
92 axs[2,1].xaxis.set_tick_params(labelsize=7)
93 axs[2,1].yaxis.set_tick_params(labelsize=7)
94
95 # Step
96 x_step = x.copy()
97 x_step[0] = 0
98
99 for epoch in range(epochs):
100     for i in range(len(iterations)-1):
101         rnd = np.random.random()
102         if rnd < 1/3:
103             x_step[epoch,i+1] = x_step[epoch,i] + (-1)
104         elif rnd < 2/3:
105             x_step[epoch,i+1] = x_step[epoch,i] + (1 - np.sqrt(3))/2
106         elif rnd < 3/3:
107             x_step[epoch,i+1] = x_step[epoch,i] + (1 + np.sqrt(3))/2
108
109     axs[1,2].plot(x_step[epoch,:], iterations, 'tab:orange', linewidth=0.3)
110
111 a = (1 - np.sqrt(3))/2
112 b = (1 + np.sqrt(3))/2
113 axs[0,2].bar([-1, a, b], [1/3, 1/3, 1/3], width=0.3, align='center',
114             color='tab:orange')
115 axs[0,2].set_xlabel('delta x')
116 axs[0,2].set_box_aspect(1)
117 axs[0,2].set_xlim([-2,2])

```

```
117 axs[0,2].set_xticks((-1, a, b))
118 axs[0,2].set_yticks(())
119 axs[0,2].xaxis.set_tick_params(labelsize=7)
120 axs[0,2].yaxis.set_tick_params(labelsize=7)
121
122 axs[1,2].set_xlabel('x(t)')
123 axs[1,2].set_box_aspect(1)
124 axs[1,2].set_xlim([-120,120])
125 axs[1,2].set_xticks((-100, -50, 0, 50, 100))
126 axs[1,2].set_yticks(())
127 axs[1,2].xaxis.set_tick_params(labelsize=7)
128 axs[1,2].yaxis.set_tick_params(labelsize=7)
129
130 axs[2,2].hist(x_step[:, -1], density=True, bins=50, color='tab:orange', alpha=0.5,
histtype='bar', ec='black')
131 axs[2,2].set_xlabel('x_1000')
132 axs[2,2].set_box_aspect(1)
133 axs[2,2].set_xlim([-120,120])
134 axs[2,2].set_xticks((-100, -50, 0, 50, 100))
135 # axs[2,2].set_yticks(())
136 axs[2,2].xaxis.set_tick_params(labelsize=7)
137 axs[2,2].yaxis.set_tick_params(labelsize=7)
138
139
140 axs[0,0].set_title('Coin flip')
141 axs[0,1].set_title('Gaussian')
142 axs[0,2].set_title('Asymmetric step')
143
144 matplotlib.rc('xtick', labels=5)
145 matplotlib.rc('ytick', labels=5)
146
147 plt.subplots_adjust(wspace=0.01, hspace=0.5)
148 plt.savefig('exercise_51ab.png', bbox_inches='tight')
149 plt.show()
```



```
1 # Exercise 52a
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6
7 fig, axs = plt.subplots(1, 3, figsize=(10,10))
8 epochs = 50
9
10 delta_t_array = np.array([0.01, 0.05, 0.1])
11 color_array = np.array(['tab:blue', 'tab:green', 'tab:orange'])
12 time_trajectory = 5
13
14 for delta_t_i in range(len(delta_t_array)):
15
16     delta_t = delta_t_array[delta_t_i]
17     no_iterations = int(time_trajectory / delta_t)
18     iterations = np.linspace(0, no_iterations, no_iterations+1)
19     time_scale = no_iterations / time_trajectory
20
21     x_t = np.zeros((epochs, no_iterations+1))
22     x_t[:,0] = 0
23
24     for epoch in range(epochs):
25         for i in range(len(iterations)-1):
26             rnd = np.random.normal()
27             x_t[epoch,i+1] = x_t[epoch,i] + (rnd * np.sqrt(delta_t))
28
29     axs[delta_t_i].plot(iterations/time_scale, x_t[epoch,:],
30 color=color_array[delta_t_i], linewidth=0.3)
31
32     axs[delta_t_i].set_xlabel('t (s)')
33     axs[delta_t_i].set_box_aspect(1)
34     axs[delta_t_i].set_ylim([-9,9])
35     axs[delta_t_i].set_xlim([0,5])
36     axs[delta_t_i].set_xticks((0, 1, 2, 3, 4, 5))
37     axs[delta_t_i].set_yticks(())
38     axs[delta_t_i].xaxis.set_tick_params(labelsize=7)
39     axs[delta_t_i].yaxis.set_tick_params(labelsize=7)
40
41 axs[0].set_yticks((-5, 0, 5))
42 axs[0].set_ylabel('x(t)')
43 axs[0].set_title('delta_t = 0.01')
44 axs[1].set_title('delta_t = 0.05')
45 axs[2].set_title('delta_t = 0.1')
46
47 matplotlib.rc('xtick', labels=5)
48 matplotlib.rc('ytick', labels=5)
49
50 plt.subplots_adjust(wspace=0.1, hspace=0.5)
51 plt.savefig('exercise_52a.png', bbox_inches='tight')
52 plt.show()
```

```

1 # Exercise 52b
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6
7 fig, axs = plt.subplots(1, 3, figsize=(10,10))
8 epochs = 10**4
9
10 delta_t_array = np.array([0.01, 0.05, 0.1])
11 color_array = np.array(['tab:blue', 'tab:green', 'tab:orange'])
12 time_trajectory = 10
13
14 for delta_t_i in range(len(delta_t_array)):
15
16     delta_t = delta_t_array[delta_t_i]
17     no_iterations = int(time_trajectory / delta_t)
18     iterations = np.linspace(0, no_iterations, no_iterations+1)
19     time_scale = no_iterations / time_trajectory
20
21     x_t = np.zeros((epochs, no_iterations+1))
22     x_t[:,0] = 0
23
24     MSD = np.zeros((epochs, len(iterations)))
25     MSD[:,0] = 0
26
27     for epoch in range(epochs):
28         for i in range(len(iterations)-1):
29             rnd = np.random.normal()
30             x_t[epoch,i+1] = x_t[epoch,i] + (rnd * np.sqrt(delta_t))
31             MSD[epoch,i+1] = np.power(x_t[epoch,i+1], 2)
32
33     MSD_average = np.zeros(len(MSD[0,:]))
34     for i in range(len(MSD_average)):
35         MSD_average[i] = np.sum(MSD[:,i]) / len(MSD[:,i])
36
37     axs[delta_t_i].plot(iterations/time_scale, MSD_average, '.', markersize=2,
38 color=color_array[delta_t_i] )
39     axs[delta_t_i].set_xlabel('t (s)')
40     axs[delta_t_i].set_box_aspect(1)
41     axs[delta_t_i].set_ylim([0,6])
42     axs[delta_t_i].set_xlim([0,5])
43     axs[delta_t_i].set_xticks((0, 1, 2, 3, 4, 5))
44     axs[delta_t_i].set_yticks(())
45     axs[delta_t_i].xaxis.set_tick_params(labelsize=7)
46     axs[delta_t_i].yaxis.set_tick_params(labelsize=7)
47
48 axs[0].set_yticks((0, 1, 2, 3, 4, 5))
49 axs[0].set_ylabel('MSD')
50 axs[0].set_title('delta_t = 0.01')
51 axs[1].set_title('delta_t = 0.05')
52 axs[2].set_title('delta_t = 0.1')
53
54 matplotlib.rc('xtick', labels=5)
55 matplotlib.rc('ytick', labels=5)
56
57 plt.subplots_adjust(wspace=0.1, hspace=0.5)
58 plt.savefig('exercise_52b.png', bbox_inches='tight')
59 plt.show()

```

```

1 # Exercise 53a
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import sys
7
8 n = 0.001
9 R = 1/10**6
10 pi = np.pi
11 gamma = 6*pi*R*n
12 T = 300
13 k = 1.380649/10**23
14 m = 1.11/10**14
15 tau = m/gamma
16 no_iterations = 100
17 dt = tau*0.01
18
19 # 1 tau long trajectory
20 timesteps = int(tau/dt)
21 iterations = np.linspace(0, 1, timesteps+1)
22 x = np.zeros(timesteps+1)
23 x_weight = x.copy()
24 x_weightless = x.copy()
25
26 for i in range(timesteps):
27     w = np.random.normal()
28     x_weight[i+1] = (x_weight[i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) -
29     (x_weight[i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m +
30     (dt*gamma)))
31     x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))
32
33 fig, axs = plt.subplots(1,2, figsize=(10,10))
34
35 axs[0].plot(iterations, x_weight, '--', color='black', label='Inertia')
36 axs[0].plot(iterations, x_weightless, color='tab:cyan', label='Non-inertia')
37 axs[0].set_xlabel('t/tau')
38 axs[0].set_ylabel('x (m)')
39 axs[0].set_box_aspect(1)
40
41 # 100 tau long trajectory
42 timesteps = int(tau/dt)*100
43 iterations = np.linspace(0, 100, timesteps+1)
44 x = np.zeros(timesteps+1)
45 x_weight = x.copy()
46 x_weightless = x.copy()
47
48 for i in range(timesteps):
49     w = np.random.normal()
50     x_weight[i+1] = (x_weight[i] * (2 + (dt*gamma/m)) / (1 + (dt*gamma/m))) -
51     (x_weight[i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w * np.sqrt(2*k*T*gamma) / (m +
52     (dt*gamma)))
53     x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))
54
55 axs[1].plot(iterations, x_weight, '--', color='black', label='Inertia')
56 axs[1].plot(iterations, x_weightless, color='tab:cyan', linewidth=0.7, label='Non-
57 inertia')
58 axs[1].set_xlabel('t/tau')

```

```
55 | axs[1].set_box_aspect(1)
56 |
57 | plt.legend(loc="upper left")
58 | plt.savefig('exercise_53a.png', bbox_inches='tight')
59 | plt.show()
```

```

1 # Exercise 53b
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import sys
7
8 n = 0.001
9 R = 1/10**6
10 pi = np.pi
11 gamma = 6*pi*R*n
12 T = 300
13 k = 1.380649/10**23
14 m = 1.11/10**14
15 tau = m/gamma
16 no_iterations = 100
17 time_step = 0.01
18 dt = tau*time_step
19
20 epochs = 10000
21 tau_trajectory = np.linspace(1,no_iterations,no_iterations)
22 x_weight_MSD = np.zeros(len(tau_trajectory))
23 x_weightless_MSD = np.zeros(len(tau_trajectory))
24 fig, ax = plt.subplots(figsize=(6,6))
25
26 for i_tau in range(len(tau_trajectory)):
27
28     timesteps = int(tau/dt)*i_tau
29     iterations = np.linspace(0, timesteps*time_step, timesteps+1)
30
31     x = np.zeros((epochs,timesteps+1))
32     x_weight = x.copy()
33     x_weightless = x.copy()
34
35     for epoch in range(epochs):
36
37         for i in range(timesteps):
38             w = np.random.normal()
39             x_weight[epoch, i+1] = (x_weight[epoch, i] * (2 + (dt*gamma/m)) / (1 +
40 (dt*gamma/m))) - (x_weight[epoch, i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w *
41 np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
42             x_weightless[epoch, i+1] = x_weightless[epoch, i] + (w *
43 np.sqrt(2*k*T*dt/gamma))
44
45         x_weight_MSD[i_tau] = np.sum(x_weight[:, -1]**2) / epochs
46         x_weightless_MSD[i_tau] = np.sum(x_weightless[:, -1]**2) / epochs
47
48     print(i_tau)
49
50 ax.plot(tau_trajectory, x_weight_MSD, '--', color='black', label='Inertia')
51 ax.plot(tau_trajectory, x_weightless_MSD, '-', color='tab:cyan', label='Non-inertia')
52 ax.set_xlabel('t/tau')
53 ax.set_ylabel('MSD (m^2)')
54 ax.set_box_aspect(1)
55 ax.set_yscale('log')
56 ax.set_xscale('log')
57 ax.set_xlim([2,100])
58 # 'linear', 'log', 'symlog', 'asinh', 'logit', 'function', 'functionlog'
59 plt.legend(loc="upper left")

```

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

```

1 # Exercise 53c
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import sys
7
8 n = 0.001
9 R = 1/10**6
10 pi = np.pi
11 gamma = 6*pi*R*n
12 T = 300
13 k = 1.380649/10**23
14 m = 1.11/10**14
15 tau = m/gamma
16 time_step = 0.01
17 dt = tau*time_step
18 tau_trajectory = 10000
19 timesteps = int(tau/dt)*tau_trajectory
20 iterations = np.linspace(0, timesteps*time_step, timesteps+1)
21 epochs = 100
22
23 # Ensemble average MSD
24 x = np.zeros((epochs,timesteps+1))
25 # x_weight = x.copy()
26 x_weightless = x.copy()
27 x_weight_ensemble_MSD = np.zeros(timesteps+1)
28 x_weightless_ensemble_MSD = np.zeros(timesteps+1)
29
30 for epoch in range(epochs):
31
32     for i in range(timesteps):
33         w = np.random.normal()
34         # x_weight[epoch,i+1] = (x_weight[epoch,i] * (2 + (dt*gamma/m)) / (1 +
35         (dt*gamma/m))) - (x_weight[epoch,i-1] / (1 + (dt*gamma/m))) + (dt**1.5 * w *
36         np.sqrt(2*k*T*gamma) / (m + (dt*gamma)))
37         x_weightless[epoch,i+1] = x_weightless[epoch,i] + (w *
38         np.sqrt(2*k*T*dt/gamma))
39
40     print(epoch)
41
42 for i in range(timesteps):
43     # x_weight_ensemble_MSD[i+1] = np.sum(x_weight[:,i]**2) / epochs
44     x_weightless_ensemble_MSD[i+1] = np.sum(x_weightless[:,i]**2) / epochs
45
46 fig, axs = plt.subplots(1,2,figsize=(10,10))
47
48 # axs[0].plot(iterations, x_weight_ensemble_MSD, '--', color='black')
49 axs[0].plot(iterations, x_weightless_ensemble_MSD, '-', color='tab:cyan')
50 axs[0].set_xlabel('t/tau')
51 axs[0].set_ylabel('MSD (m^2)')
52 axs[0].set_title('Ensemble averaged MSD')
53 axs[0].set_box_aspect(1)
54
55 # Time average MSD
56 x = np.zeros((timesteps+1))
57 x_weight = x.copy()
58 x_weightless = x.copy()
59 x_weight_time_MSD = np.zeros(timesteps+1)

```

```
57 x_weightless_time_MSD = np.zeros(timesteps+1)
58
59 for i in range(timesteps):
60     w = np.random.normal()
61     # 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)))
62     x_weightless[i+1] = x_weightless[i] + (w * np.sqrt(2*k*T*dt/gamma))
63
64     # x_weight_time_MSD[i+1] = x_weight[i]**2 / timesteps
65     x_weightless_time_MSD[i+1]= x_weightless[i]**2 / timesteps
66
67     if i % 1000 == 0:
68         print(i)
69
70 # axs[1].plot(iterations, x_weight_time_MSD, '--', color='black', label='Inertia')
71 axs[1].plot(iterations, x_weightless_time_MSD, '-', color='tab:cyan', label='Non-
inertia')
72 axs[1].set_xlabel('t/tau')
73 axs[1].set_box_aspect(1)
74 axs[1].set_title('Time averaged MSD')
75
76 # plt.legend(loc="upper left")
77 plt.savefig('exercise_53c.png', bbox_inches='tight')
78 plt.show()
```



```

1 # Exercise 54ab
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import sys
7
8 n = 0.001
9 R = 1/10**6
10 gamma = 6*np.pi*n*R
11 k = 1.380649/10**23
12 T = 300
13 kx = (1/10**12) / (1/10**6)
14 ky = (0.25/10**12) / (1/10**6)
15 tau = gamma / kx # Arbitrary choose kx
16 time_step = 0.01
17 dt = tau*time_step
18 tau_trajectory = 10000
19 timesteps = int(tau/dt)*tau_trajectory
20 iterations = np.linspace(0, timesteps*time_step, timesteps+1)
21 x = np.zeros(timesteps+1)
22 y = np.zeros(timesteps+1)
23
24 for i in range(timesteps-1):
25     w = np.random.normal()
26     x[i+1] = x[i] - (dt*x[i]*kx/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
27     w = np.random.normal()
28     y[i+1] = y[i] - (dt*y[i]*ky/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
29
30 xmax = 5/10**7
31 ymax = 7*10**6
32 x_sym = np.linspace(-xmax, xmax, timesteps)
33 y_sym = x_sym
34 px = 6*10**6*np.exp(-(0.5*kx*x_sym**2)/(k*T))
35 py = 3*10**6*np.exp(-(0.5*ky*y_sym**2)/(k*T))
36
37 fig, axs = plt.subplots(1,3,figsize=(12,12))
38
39 axs[0].plot(x, y, '.', color='tab:blue', markersize=0.1)
40 axs[0].set_xlabel('x (m)')
41 axs[0].set_ylabel('y (m)')
42 axs[0].set_xlim([-xmax, xmax])
43 axs[0].set_ylim([-xmax, xmax])
44 axs[0].set_box_aspect(1)
45
46 weights_x = 30*np.ones_like(x)/len(x)
47 weights_y = 30*np.ones_like(y)/len(y)
48
49 axs[1].hist(x, density=True, bins=100, color='tab:blue', alpha=0.5, label='Brownian')
50 axs[1].plot(x_sym, px, '--', color='red', label='Theory')
51 axs[1].set_xlabel('x (m)')
52 axs[1].set_ylabel('p(x)')
53 axs[1].set_box_aspect(1)
54 axs[1].set_xlim([-xmax, xmax])
55 axs[1].legend(loc="upper left",prop={'size': 8})
56 axs[1].set_ylim([0, ymax])
57 axs[1].set_yticks(())
58

```

```
59 axs[2].hist(y, density=True, bins=100, color='tab:green', alpha=0.5,  
label='Brownian')  
60 axs[2].plot(y_sym, py, '--', color='red', label='Theory')  
61 axs[2].set_xlabel('y (m)')  
62 axs[2].set_ylabel('p(y)')  
63 axs[2].set_box_aspect(1)  
64 axs[2].set_xlim([-xmax, xmax])  
65 axs[2].legend(loc="upper left",prop={'size': 8})  
66 axs[2].set_ylim([0, ymax])  
67 axs[2].set_yticks(())  
68  
69 plt.subplots_adjust(wspace=0.3, hspace=0.5)  
70 plt.savefig('exercise_54ab.png', bbox_inches='tight')  
71 plt.show()  
72
```

```

1 # Exercise 54c
2
3 import numpy as np
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import sys
7
8 n = 0.001
9 R = 1/10**6
10 gamma = 6*np.pi*n*R
11 k = 1.380649/10**23
12 T = 300
13 kx = (1/10**12) / (1/10**6)
14 ky = (0.25/10**12) / (1/10**6)
15 tau = gamma / kx # Arbitrary choose kx
16 time_step = 0.001
17 dt = tau*time_step
18 tau_trajectory = 10
19 timesteps = int(tau/dt)*tau_trajectory
20 iterations = np.linspace(0, timesteps*time_step, timesteps+1)
21 x = np.zeros(timesteps+1)
22 y = np.zeros(timesteps+1)
23 epochs = 10
24 cx = np.zeros((epochs,timesteps+1))
25 cy = np.zeros((epochs,timesteps+1))
26 cx_average = np.zeros(timesteps+1)
27 cy_average = np.zeros(timesteps+1)
28
29 for epoch in range(epochs):
30     for i in range(timesteps-1):
31         w = np.random.normal()
32         x[i+1] = x[i] - (dt*x[i]*kx/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
33         w = np.random.normal()
34         y[i+1] = y[i] - (dt*y[i]*ky/gamma) + (w*np.sqrt(2*k*T*dt/gamma))
35
36     print(timesteps)
37     for i in range(timesteps):
38         if i % 100 == 0:
39             print(i)
40             for j in range(timesteps-i):
41                 cx[epoch,j] = cx[epoch,j] + x[i]*x[i+j]
42                 cy[epoch,j] = cy[epoch,j] + y[i]*y[i+j]
43             cx[epoch,i] = cx[epoch,i]/(timesteps-i)
44             cy[epoch,i] = cy[epoch,i]/(timesteps-i)
45     cx[epoch,:] = cx[epoch,:]/timesteps
46     cy[epoch,:] = cy[epoch,:]/timesteps
47
48     print(epoch)
49
50 for i in range(timesteps):
51     cx_average[i] = np.sum(cx[:,i]) / epochs
52     cy_average[i] = np.sum(cy[:,i]) / epochs
53
54 t = np.linspace(0, 17*tau, timesteps+1)
55 cx_theory = (k*T/kx)*np.exp(-(kx*t/gamma))
56 cy_theory = (k*T/ky)*np.exp(-(ky*t/gamma))
57
58 fig, ax = plt.subplots(1,2,figsize=(10,10))
59

```

```
60 ax[0].plot(t, cx_average, '-', color='tab:blue', label='Autocorrelation')
61 ax[0].plot(t, cx_theory, '--', color='red', label='Theory')
62 ax[0].set_xlabel('t (s)')
63 ax[0].set_ylabel('C_x')
64 ymax = 2/10**14
65 ymin = -1/10**15
66 ax[0].set_ylim([ymin,ymax])
67 ax[0].legend(loc="upper right",prop={'size': 8})
68 ax[0].set_box_aspect(1)
69 ax[0].set_yticks(())
70
71 ax[1].plot(t, cy_average, '-', color='tab:green', label='Autocorrelation')
72 ax[1].plot(t, cy_theory, '--', color='red', label='Theory')
73 ax[1].set_xlabel('t (s)')
74 ax[1].set_ylabel('C_y')
75 ax[1].set_yticks(())
76 ax[1].set_ylim([ymin,ymax])
77 ax[1].legend(loc="upper right",prop={'size': 8})
78 ax[1].set_box_aspect(1)
79
80 plt.subplots_adjust(wspace=0.3, hspace=0.5)
81 plt.savefig('exercise_54c.png', bbox_inches='tight')
82 plt.show()
83
```