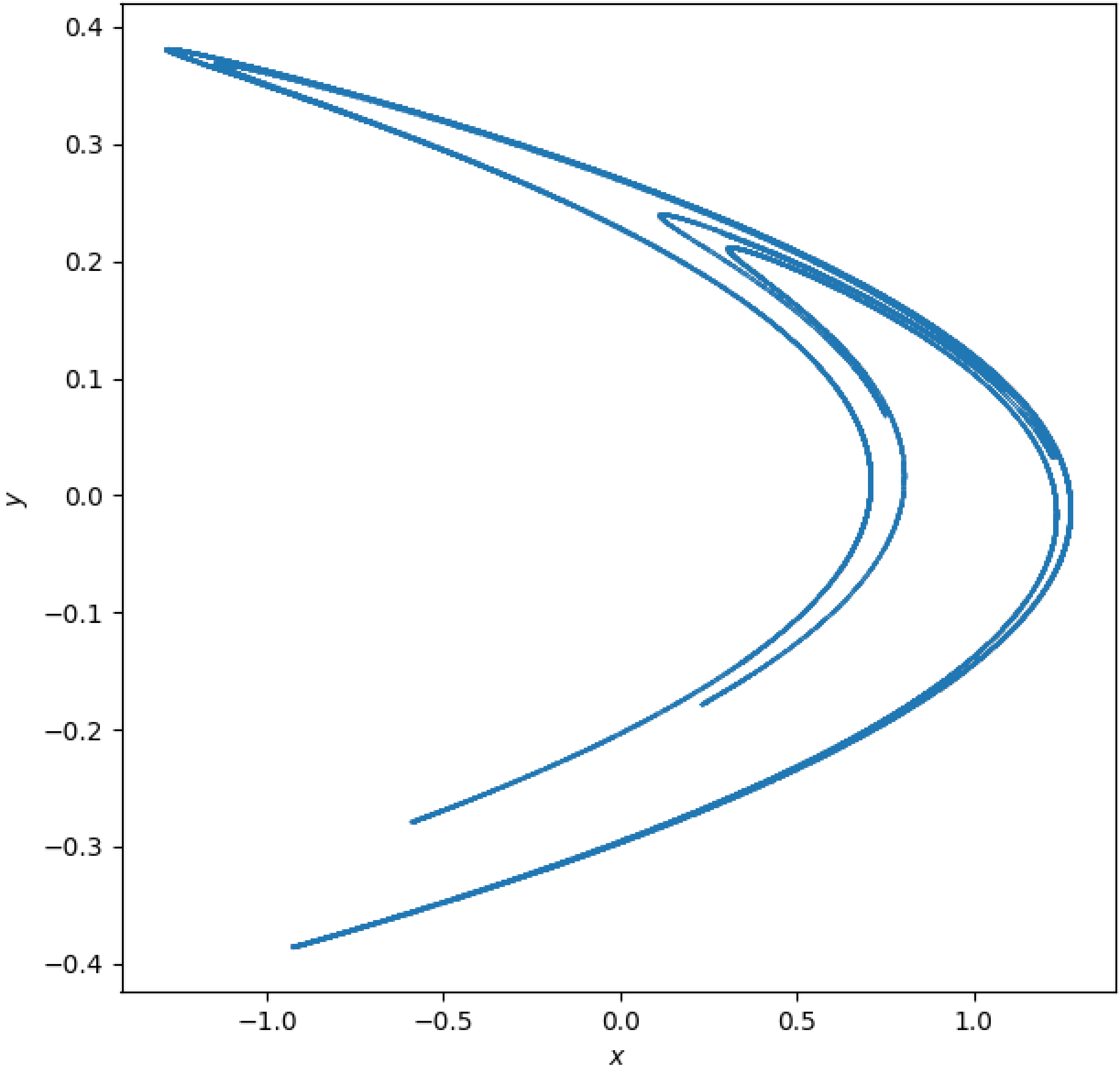
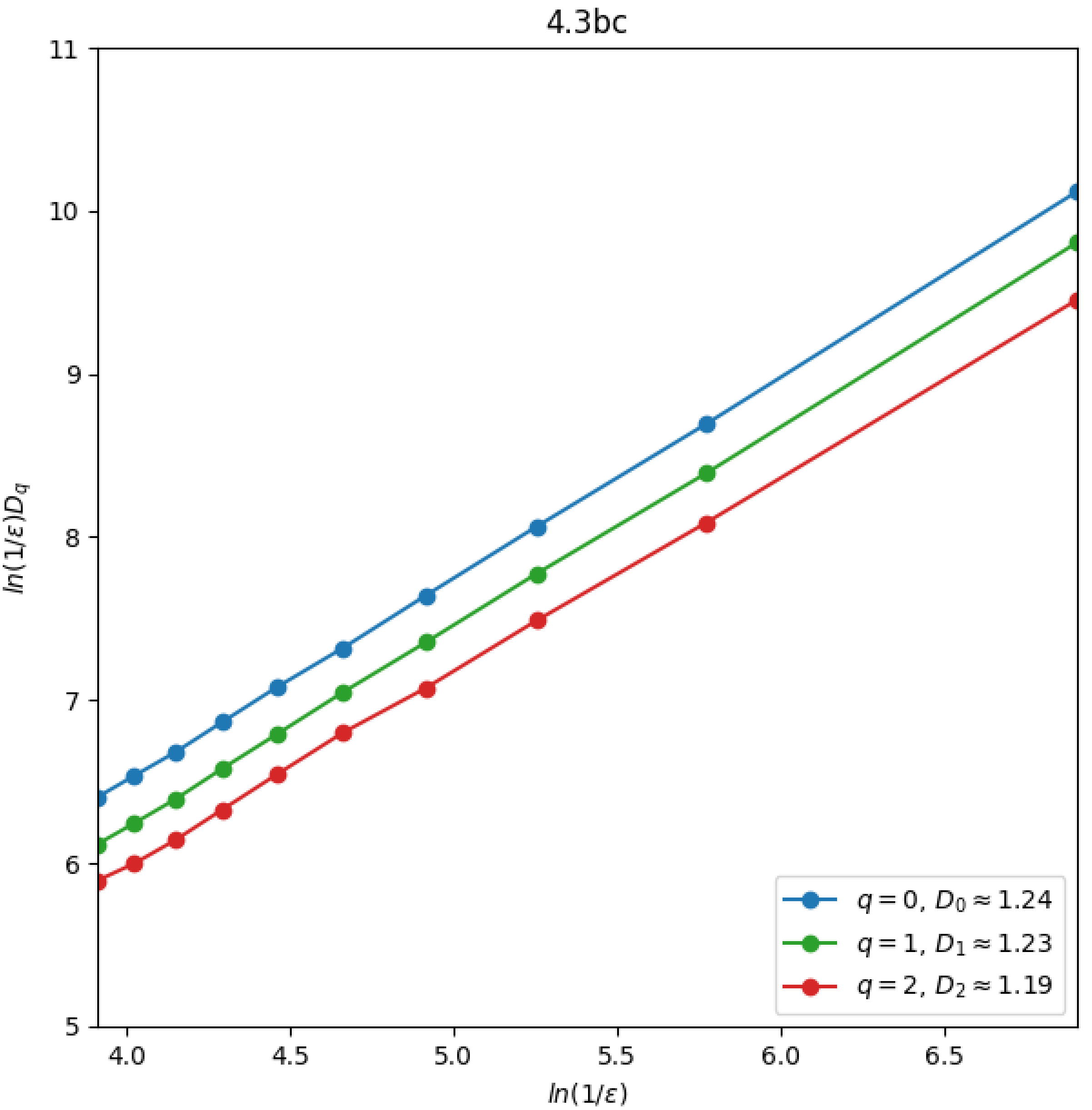
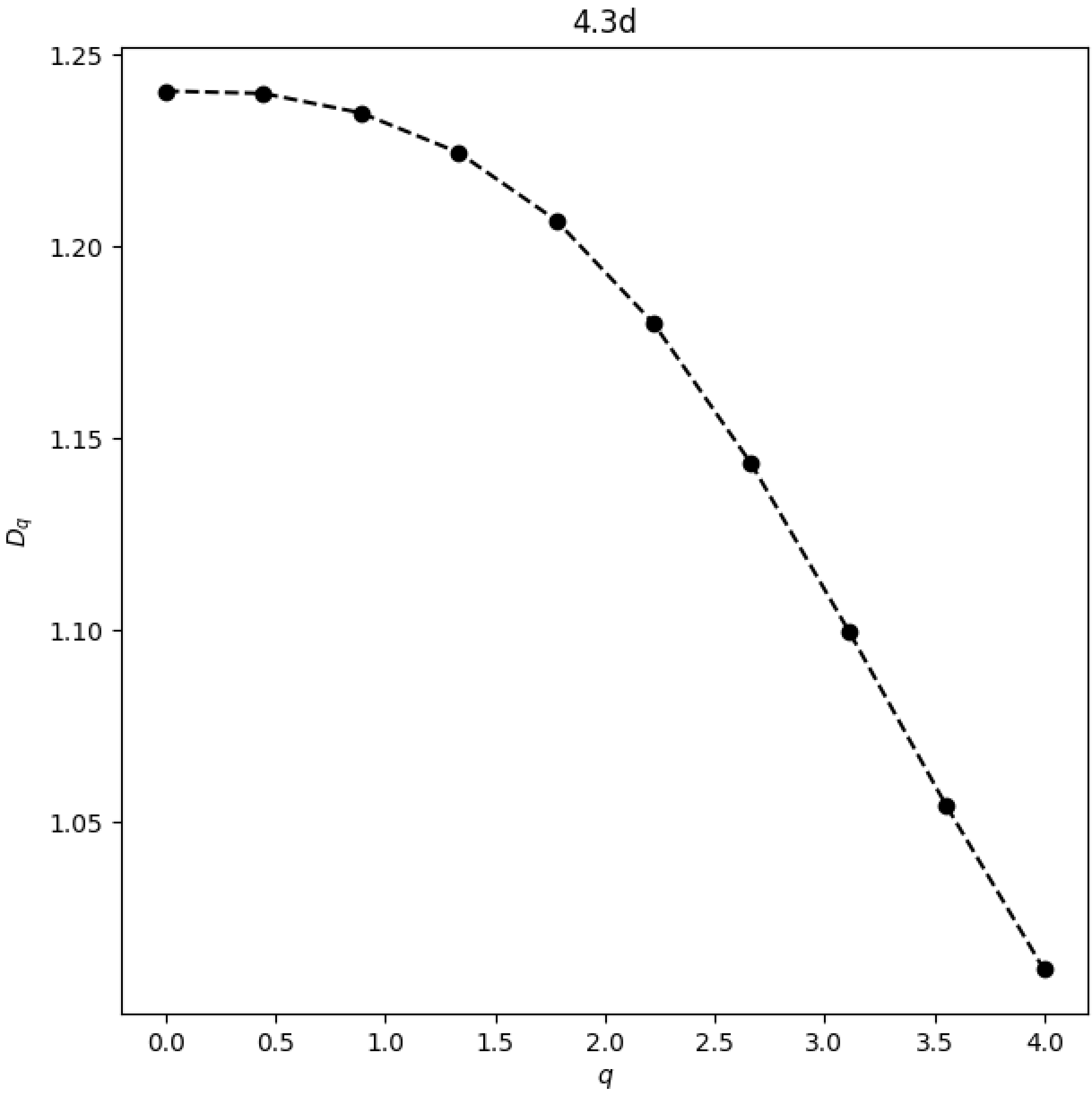
4.3a




4.3bc
$ln(1/\varepsilon)D_q$
$ln(1/\varepsilon)$
11
10
9
8
7
6
5
4.0
4.5
5.0
5.5
6.0
6.5
$q=0$, $D_0 \approx 1.24$
$q=1$, $D_1 \approx 1.23$
$q=2$, $D_2 \approx 1.19$


4.3d
D_q
q
1.25
1.20
1.15
1.10
1.05
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```python
# Exercise 4.3a

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

title = '4.3a'

a = 1.4
b = 0.3

cut_tail = 100

T = 1000
dt = 0.01
t = np.arange(0,T,dt)

x = np.zeros_like(t,dtype=float)
y = x.copy()


x[0] = (np.random.uniform()-0.5)
y[0] = (np.random.uniform()-0.5)

for t in range(len(x)-1):
    x[t+1] = y[t]+1-a*x[t]**2
    y[t+1] = b*x[t]

plt.figure(figsize=(7,7))
plt.plot(x[cut_tail:],y[cut_tail:],'.',markersize=0.2)
plt.title(title)
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.savefig('Dynamical Systems/DS HW4/4.3/'+title+'.png')
plt.show()
```

```python
# Exercise 4.3bc

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

title = '4.3bc'

a = 1.4
b = 0.3

cut_tail = 100

T = 10**4
dt = 5*10**-3
t = np.arange(0,T,dt)

x = np.zeros_like(t,dtype=float)
y = x.copy()

x[0] = (np.random.uniform()-0.5)
y[0] = (np.random.uniform()-0.5)

for t in range(len(x)-1):
    x[t+1] = y[t]+1-a*x[t]**2
    y[t+1] = b*x[t]

x = x[cut_tail:]
y = y[cut_tail:]

q_vals = np.linspace(0,2,3)
epsilon_range = np.linspace(10**-3, 2*10**-2, 10)
fig1, axs = plt.subplots(1,3,figsize=(15,15))
fig2, ax = plt.subplots(figsize=(7,7))
color = ['tab:blue', 'tab:green', 'tab:red']

for q_i in range(len(q_vals)):

    q = q_vals[q_i]
    Iq = np.zeros_like(epsilon_range, dtype=float)

    for epsilon_i in range(len(epsilon_range)):

        epsilon = epsilon_range[epsilon_i]
        N_points = len(x)

        xmax = 1.3
        xmin = -xmax
        ymax = 0.4
        ymin = -ymax

        x_bins = np.linspace(xmin, xmax, int((xmax-xmin)/epsilon))
        y_bins = np.linspace(ymin, ymax, int((ymax-ymin)/epsilon))

        plt.figure()
        histogram = plt.hist2d(x, y, bins=[x_bins, y_bins])
        boxes = histogram[0].copy()
        plt.figure().clear()
        plt.close()
        plt.cla()
        plt.clf()

        for i in range(len(boxes[:,0])):
            for j in range(len(boxes[0,:])):
                if boxes[i,j] != 0:
                    if q == 1:
                        N_k = boxes[i,j]
                        Iq[epsilon_i] += ((N_k/N_points)*np.log(1/(N_k/N_points)))
                    else:
                        N_k = boxes[i,j]
                        Iq[epsilon_i] += (N_k/N_points)**q


    x_axis = np.log(1/epsilon_range)
    axs[q_i].set_xlabel(r'$ln(1/\epsilon)$')
    ax.set_xlabel(r'$ln(1/\epsilon)$')
    ax.set_ylabel(r'$ln(1/\epsilon) D_{q}$')

```

```
    if q == 1:
        y_axis = Iq
        axs[q_i].set_ylabel(r'$\sum_{k}^{N_{box}} p_{k} ln(1/p_{k})$')
    else:
        y_axis = np.log(Iq)/(1-q)
        axs[q_i].set_ylabel(r'$ln(\sum_{k}^{N_{box}} p_{k}^{q})/(1-q)$')

    coef = np.polyfit(x_axis, y_axis, 1)
    print(coef)
    Dq = coef[0]

    axs[q_i].plot(x_axis, y_axis, color=color[q_i])
    axs[q_i].set_title('$q={}$, $D_{}={}$'.format(int(q),int(q),np.round(Dq,2)))
    axs[q_i].set_box_aspect(1)
    axs[q_i].set_ylim([5,11])
    axs[q_i].set_xlim([np.log(1/epsilon_range[-1]), np.log(1/epsilon_range[0])])

    legend = '$q={}$, $D_{}'.format(int(q),int(q)) + r'\approx' + '{}$'.format(np.round(Dq,2))
    ax.plot(x_axis, y_axis, 'o-', color=color[q_i], label=legend)
    ax.set_title(title)
    ax.set_box_aspect(1)
    ax.set_ylim([5,11])
    ax.set_xlim([np.log(1/epsilon_range[-1]), np.log(1/epsilon_range[0])])

plt.subplots_adjust(wspace=1)
ax.legend(loc='lower right')
fig1.savefig('Dynamical Systems/DS HW4/4.3/'+title+'_v1.png')
fig2.savefig('Dynamical Systems/DS HW4/4.3/'+title+'_v2.png')
plt.show()
```

```python
# Exercise 4.3d

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

title = '4.3d'

a = 1.4
b = 0.3

cut_tail = 100

T = 10**4
dt = 5*10**-3
t = np.arange(0,T,dt)

x = np.zeros_like(t,dtype=float)
y = x.copy()

x[0] = (np.random.uniform()-0.5)
y[0] = (np.random.uniform()-0.5)

for t in range(len(x)-1):
    x[t+1] = y[t]+1-a*x[t]**2
    y[t+1] = b*x[t]

x = x[cut_tail:]
y = y[cut_tail:]

q_vals = np.linspace(0,4,10)
Dq_list = []
epsilon_range = np.linspace(10**-3, 2*10**-2, 10)
fig, ax = plt.subplots(figsize=(7,7))
color = ['tab:blue', 'tab:green', 'tab:red']

for q_i in range(len(q_vals)):

    q = q_vals[q_i]
    Iq = np.zeros_like(epsilon_range, dtype=float)

    for epsilon_i in range(len(epsilon_range)):

        epsilon = epsilon_range[epsilon_i]
        N_points = len(x)

        xmax = 1.3
        xmin = -xmax
        ymax = 0.4
        ymin = -ymax

        x_bins = np.linspace(xmin, xmax, int((xmax-xmin)/epsilon))
        y_bins = np.linspace(ymin, ymax, int((ymax-ymin)/epsilon))

        plt.figure()
        histogram = plt.hist2d(x, y, bins=[x_bins, y_bins])
        boxes = histogram[0].copy()
        plt.figure().clear()
        plt.close()
        plt.cla()
        plt.clf()

        for i in range(len(boxes[:,0])):
            for j in range(len(boxes[0,:])):
                if boxes[i,j] != 0:
                    if q == 1:
                        N_k = boxes[i,j]
                        Iq[epsilon_i] += ((N_k/N_points)*np.log(1/(N_k/N_points)))
                    else:
                        N_k = boxes[i,j]
                        Iq[epsilon_i] += (N_k/N_points)**q

    x_axis = np.log(1/epsilon_range)

    if q == 1:
        y_axis = Iq
    else:
        y_axis = np.log(Iq)/(1-q)
```

```python

    coef = np.polyfit(x_axis, y_axis, 1)
    Dq = coef[0]
    Dq_list.append(Dq)
    print('Dq=',Dq,'    q=',q)

ax.plot(q_vals, Dq_list, 'o--', color='black')
ax.set_title(title)
ax.set_box_aspect(1)
ax.set_xlabel(r'$q$')
ax.set_ylabel(r'$D_{q}$')

fig.savefig('Dynamical Systems/DS HW4/4.3/'+title+'.png')
plt.show()
```

```python
# Exercise 4.3ef

import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import qr
import sys

title = '4.3e'

a = 1.4
b = 0.3

cut_tail = 100

T = 2*10**5
t_array = np.arange(0,T,1)

x = np.zeros_like(t_array,dtype=float)
y = x.copy()

x[0] = (np.random.uniform()-0.5)
y[0] = (np.random.uniform()-0.5)

for t in range(len(x)-1):
    x[t+1] = y[t]+1-a*x[t]**2
    y[t+1] = b*x[t]

x = x[cut_tail:]
y = y[cut_tail:]

I = np.identity(2)
Q = I.copy()
M = I.copy()

lambda_one = np.zeros_like(x)
lambda_two = lambda_one.copy()

new_T = len(x)
new_t_array = t_array[cut_tail:]-cut_tail

for t in range(len(x)):

    J11 = 2*a*x[t]
    J12 = 1
    J21 = b
    J22 = 0

    J = np.array([[J11,J12],\
                  [J21,J22]])

    M = J
    Q,R = qr(np.matmul(M,Q))

    lambda_one[t] = lambda_one[t-1] + np.log(np.absolute(R[0,0]))/new_T
    lambda_two[t] = lambda_two[t-1] + np.log(np.absolute(R[1,1]))/new_T

lambda_one_converged = lambda_one[-1]
lambda_two_converged = lambda_two[-1]

if lambda_one_converged < lambda_two_converged:
    temp = lambda_one_converged
    lambda_one_converged = lambda_two_converged
    lambda_two_converged = temp

lyapunov_dimension = 1 + lambda_one_converged/np.abs(lambda_two_converged)

print('Lyapunov exponent 1: ', lambda_one_converged)
print('Lyapunov exponent 2: ', lambda_two_converged)
print('Lyapunov dimension: ', lyapunov_dimension)

fig2, ax2 = plt.subplots(figsize=(7,7))
ax2.plot(new_t_array[1:], np.divide(lambda_one[1:],new_t_array[1:]), '-',
linewidth=1.5, label=r'$\lambda_1$ (conv. at $\approx$' +
'{})'.format(np.round(lambda_one_converged,2)))
ax2.plot(new_t_array[1:], np.divide(lambda_two[1:],new_t_array[1:]), '-',
linewidth=1.5, label=r'$\lambda_2$ (conv. at $\approx$' +
'{})'.format(np.round(lambda_two_converged,2)))
ax2.set_xscale('log')
```

```
ax2.set_xlabel('$t$')
ax2.set_ylabel('$\sum \lambda_i$ /t')
ax2.set_box_aspect(1)
ax2.set_title(title)

plt.legend(loc="lower right", prop={'size': 10})
plt.savefig('Dynamical Systems/DS HW4/4.3/'+title+'.png')
plt.show()
```