HOMEWORK SET 1

ΓΑΛΑΝΗΣ ΑΧΙΛΛΕΑΣ ΑΛΕΞΑΝΔΡΟΣ ΒΑΣΙΛΕΙΟΣ - 02941 and ΓΑΛΑΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΟΡΕΣΤΗΣ ΒΑΣΙΛΕΙΟΣ - 03074

Νευρο-Ασαφής Υπολογιστική 2023-24
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας, Βόλος
{acgalanis, kogalanis}@uth.gr

Περιεχόμενα

HOMEWORK SET 1 1

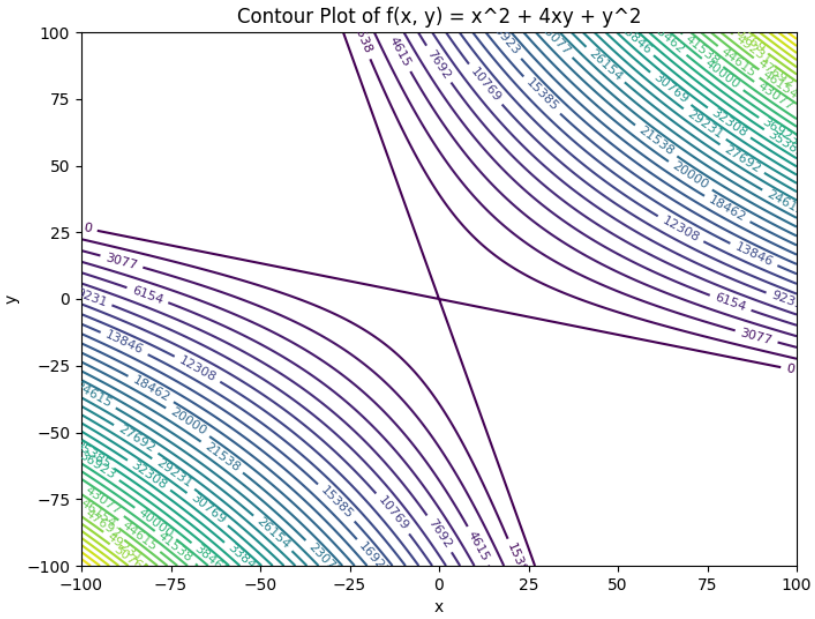
ΓΑΛΑΝΗΣ ΑΧΙΛΛΕΑΣ ΑΛΕΞΑΝΔΡΟΣ ΒΑΣΙΛΕΙΟΣ - 02941 and
ΓΑΛΑΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΟΡΕΣΤΗΣ ΒΑΣΙΛΕΙΟΣ - 03074

1 PROBLEM-01 2
2 PROBLEM-02 4
3 PROBLEM-03 6
4 PROBLEM-04 47
5 PROBLEM-05 49
6 PROBLEM-06 52
7 PROBLEM-07 52
8 PROBLEM-08 57
9 PROBLEM-09 61
10 PROBLEM-10 65
11 PROBLEM-11 69
12 PROBLEM-12 70

1 PROBLEM-01

– **Plot Contour Lines:** For the function $f(x, y) = x^2 + 4xy + y^2$ the contour lines are defined by setting the function equal to a constant **k**: $x^2 + 4xy + y^2 = k$.
Let's consider a few specific values for k to demonstrate how to calculate the contour lines. For instance, for **k = 0, 10, 20:**

- **For k = 0:** $x^2 + 4xy + y^2 = 0 \Rightarrow y^2 + 4xy + x^2 = 0$. This is a quadratic equation in terms of y. Using the quadratic formula: $y = \frac{-4x \pm \sqrt{16x^2 - 4x^2}}{2}$ thus, the contour line is: $y = \frac{-4x \pm \sqrt{16x^2 - 4x^2}}{2}$
- **For k = 10:** $x^2 + 4xy + y^2 = 10 \Rightarrow y^2 + 4xy + x^2 - 10 = 0$. This is a quadratic equation in terms of y. Using the quadratic formula: $y = \frac{-4x \pm \sqrt{16x^2 - 4(x^2 - 10)}}{2}$ thus, the contour line is: $y = \frac{-4x \pm \sqrt{16x^2 - 4(x^2 - 10)}}{2}$
- **For k = 20:** $x^2 + 4xy + y^2 = 20 \Rightarrow y^2 + 4xy + x^2 - 20 = 0$. This is a quadratic equation in terms of y. Using the quadratic formula: $y = \frac{-4x \pm \sqrt{16x^2 - 4(x^2 - 20)}}{2}$ thus, the contour line is: $y = \frac{-4x \pm \sqrt{16x^2 - 4(x^2 - 20)}}{2}$



– **Quadratic Form:** To express the function $f(x, y) = x^2 + 4xy + y^2$ in quadratic form, we can represent it using matrix notation. In quadratic form, a function of two variables is typically expressed as: $f(z) = z^T Az + b^T z + c$, where z is a column vector of variables, A is a symmetric matrix, b is a column vector, and c is a scalar.For the given function, there are no linear or constant terms (i.e. b = 0 and c = 0), so the quadratic form is simplified to: $f(z) = z^T Az$.
For $f(x, y) = x^2 + 4xy + y^2$, let's define:
$z = \begin{bmatrix} x \\ y \end{bmatrix}$.
The symmetric matrix A can be determined by comparing the given function with the general form. The function $x^2 + 4xy + y^2$ corresponds to:
$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$.
So, the quadratic form of the function is:
$f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

– **Local Minima/Maxima:** The first step to finding the critical points is to calculate the gradient of the function. The gradient is:
$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + 4y \\ 4x + 2y \end{bmatrix}$.
Then we solve the system of equations obtained by setting the gradient equal to zero:

$$2x + 4y = 0 \quad (1)$$
$$4x + 2y = 0 \quad (2)$$

.
The only critical point is at (x,y)=(0,0). To characterize the critical point, we use the second derivative test. We compute the Hessian matrix (H) :
$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}$.

- **Second Partial Derivative with Respect to x:** $\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(2x + 4y) = 2$.
- **Mixed Partial Derivative (x then y):** $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial y}(2x + 4y) = 4$.
- **Mixed Partial Derivative (y then x):** $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial x}(4x + 2y) = 4$.
- **Second Partial Derivative with Respect to y:** $\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(4x + 2y) = 2$.

(Note: Since f(x,y) is continuous because it is polynomial function and has continuous partial derivatives, the mixed partial derivatives are equal by Clairaut's theorem).
Then, we find the determinant of the Hessian matrix:

$$\begin{aligned} \text{Det}(H) &= \text{Det}\left(\begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}\right) \\ &= (2 \times 2) - (4 \times 4) \\ &= 4 - 16 \\ &= -12 \end{aligned}$$

.
Finally:
If $D > 0$ and $\frac{\partial^2 f}{\partial x^2} > 0$, there is a local minimum.
If $D > 0$ and $\frac{\partial^2 f}{\partial x^2} < 0$, there is a local maximum.
If $D < 0$, there is a saddle point.
The $\frac{\partial^2 f}{\partial x^2}$ is equal to 2 and Det(H) = -12. Since Det(H) < 0 and $\frac{\partial^2 f}{\partial x^2} > 0$, the function does not have local Minima/Maxima but it has a saddle point at (0,0).

2 PROBLEM-02

The Gradient Descent algorithm is an iterative method for finding the minimum of a function. Here's the basic formula for updating the variables: $x_{n+1} = x_n - \lambda \frac{\nabla f(x_n)}{\|\nabla f(x_n)\|}$, where λ is the learning rate, $\nabla f(x_n)$ is the gradient of the function at the current point and $\|\nabla f(x_n)\|$ is the L2 norm of the gradient at the current point.
Given: $f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$, we need to calculate its gradient:
$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$.

We find the gradient of the function:
$\frac{\partial f}{\partial x_1} = \frac{\partial((x_1+2x_2-7)^2+(2x_1+x_2-5)^2)}{\partial x_1} = 10x_1 + 8x_2 - 34.$
$\frac{\partial f}{\partial x_2} = \frac{\partial((x_1+2x_2-7)^2+(2x_1+x_2-5)^2)}{\partial x_2} = 8x_1 + 10x_2 - 38.$
We calculate the norm of the gradient of the function:
$\|\nabla(f(x_1, x_2)\| = \sqrt{(8x_1 + 10x_2 - 38)^2 + (10x_1 + 8x_2 - 34)^2}$

- **Iteration 1:** Thus, for $x_0 = (-9.5, 9.5)$ we calculate: $\nabla f(-9.5, 9.5) = \begin{bmatrix} -53 \\ -19 \end{bmatrix}$ and $\|\nabla(f(-9.5, 9.5)\| = \sqrt{3170}$.

  Now, we compute the direction of descending: $-\frac{\nabla f(x_0)}{\|\nabla(f(x_0)\|} = -\frac{1}{\sqrt{3170}} \times \begin{bmatrix} -53 \\ -19 \end{bmatrix}$.

  So, for $x_0 = (-9.5, 9.5)$: $x_1 = \begin{bmatrix} -9.5 \\ 9.5 \end{bmatrix} + \lambda \times (-\frac{1}{\sqrt{3170}}) \times \begin{bmatrix} -53 \\ -19 \end{bmatrix} =$
  $(-9.5 + \frac{53 \times \lambda}{\sqrt{3170}}, 9.5 + \frac{19 \times \lambda}{\sqrt{3170}})$.
  To find the optimal λ we need to substitute x_1 into the function f differentiate $f(x_1)$ with respect to λ and set the derivative to zero afterwards. The function $f(x_1)$ in terms of λ becomes:
  $f(x_1) = f(-9.5 + \frac{53 \times \lambda}{\sqrt{3170}}, 9.5 + \frac{19 \times \lambda}{\sqrt{3170}}) = ((-9.5 + \frac{53 \times \lambda}{\sqrt{3170}}) + 2 \times (9.5 + \frac{19 \times \lambda}{\sqrt{3170}}) - 7)^2 + (2 \times (-9.5 + \frac{53 \times \lambda}{\sqrt{3170}})) + (9.5 + \frac{19 \times \lambda}{\sqrt{3170}}) - 5)^2) =$
  $(2.5 + \frac{91 \times \lambda}{\sqrt{3170}})^2 + (-14.5 + \frac{125 \times \lambda}{\sqrt{3170}})^2$.
  We differentiate $f(x_1)$ with respect to λ: $\frac{d}{d\lambda} f(x_1) = 2 \times (2.5 + \frac{91 \times \lambda}{\sqrt{3170}}) \times (\frac{91}{\sqrt{3170}}) + 2 \times (-14.5 + \frac{125 \times \lambda}{\sqrt{3170}}) \times (\frac{125}{\sqrt{3170}})$.
  We set the derivative to 0 to find the optimal λ: $\frac{d}{d\lambda} f(x_1) = 0 \Rightarrow 2 \times (2.5 + \frac{91 \times \lambda}{\sqrt{3170}}) \times (\frac{91}{\sqrt{3170}}) + 2 \times (-14.5 + \frac{125 \times \lambda}{\sqrt{3170}}) \times (\frac{125}{\sqrt{3170}}) = 0 \Rightarrow (\frac{455}{\sqrt{3170}} + \frac{16562 \times \lambda}{3170}) + (-\frac{3625}{\sqrt{3170}} + \frac{31250 \times \lambda}{3170}) = 0 \Rightarrow \frac{47812 \times \lambda}{3170} = \frac{3170}{\sqrt{3170}} \Rightarrow \lambda \approx 3.7329$.

  Now, let's calculate x_1:
  $x_1 = (-9.5 + \frac{53 \times 3.7329}{\sqrt{3170}}, 9.5 + \frac{19 \times 3.7329}{\sqrt{3170}}) \approx (-5.986, 10.759)$.

- **Iteration 2:** Thus, for $x_1 = (-5.986, 10.759)$ we calculate: $\nabla f(-5.986, 10.759) = \begin{bmatrix} -7.788 \\ 21.702 \end{bmatrix}$ and $\|\nabla(f(-5.986, 10.759)\| \approx \sqrt{151.588}$.

  Now, we compute the direction of descending: $-\frac{\nabla f(x_1)}{\|\nabla(f(x_1)\|} = -\frac{1}{\sqrt{151.588}} \times \begin{bmatrix} -7.788 \\ 21.702 \end{bmatrix}$.

  So, for $x_1 = (-5.986, 10.759)$: $x_2 = \begin{bmatrix} -5.986 \\ 10.759 \end{bmatrix} + \lambda \times (-\frac{1}{\sqrt{151.588}}) \times \begin{bmatrix} -7.788 \\ 21.702 \end{bmatrix} =$
  $(-5.986 + \frac{7.788 \times \lambda}{\sqrt{151.588}}, 10.759 - \frac{21.702 \times \lambda}{\sqrt{151.588}})$.

  To find the optimal λ we need to substitute x_2 into the function f differentiate $f(x_2)$ with respect to λ and set the derivative to zero afterwards. The function $f(x_2)$ in terms of λ becomes:
  $f(x_2) = f(-5.986 + \frac{7.788 \times \lambda}{\sqrt{151.588}}, 10.759 - \frac{21.702 \times \lambda}{\sqrt{151.588}}) = ((-5.986 + \frac{7.788 \times \lambda}{\sqrt{151.588}}) +$

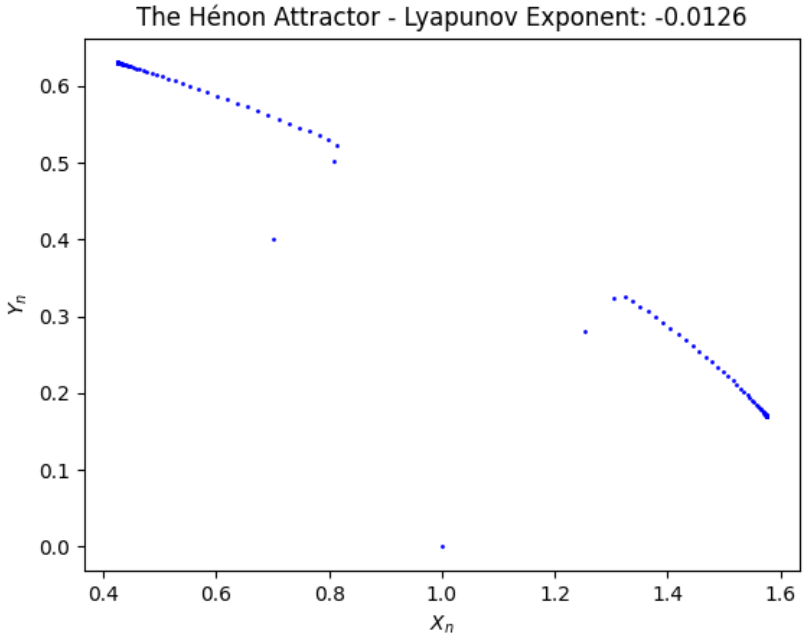
$2\times(10.759-\frac{21.702\times\lambda}{\sqrt{151.588}})-7)^2+(2\times(-5.986+\frac{7.788\times\lambda}{\sqrt{151.588}})+(10.759-\frac{21.702\times\lambda}{\sqrt{151.588}})-5)^2=(8.532-\frac{35.616\times\lambda}{\sqrt{151.588}})^2+(-6.213-\frac{6.126\times\lambda}{\sqrt{151.588}})$.
We differentiate $f(x_2)$ with respect to λ: $\frac{d}{d\lambda}f(x_2)=2\times(8.532-\frac{35.616\times\lambda}{\sqrt{151.588}})\times(-\frac{35.616}{\sqrt{151.588}})+2\times(-6.213-\frac{6.126\times\lambda}{\sqrt{151.588}})\times(-\frac{6.126}{\sqrt{151.588}})$.
We set the derivative to 0 to find the optimal λ: $\frac{d}{d\lambda}f(x_2)=0\Rightarrow(-\frac{607.751}{\sqrt{151.588}}+\frac{2536.998\times\lambda}{151.588})+(\frac{76.121}{\sqrt{151.588}}+\frac{75.055\times\lambda}{151.588})=0\Rightarrow\frac{2612.053\times\lambda}{151.588}=\frac{532.696}{\sqrt{151.588}}\Rightarrow\lambda\approx2.510$.

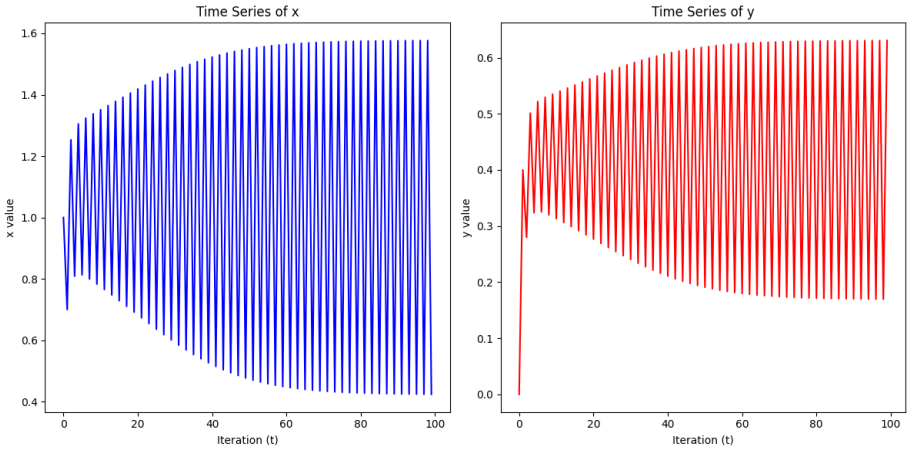
Now, let's calculate x_2:
$x_2=(-5.986+\frac{7.788\times2.510}{\sqrt{151.588}},10.759-\frac{21.702\times2.510}{\sqrt{151.588}})=(-4.398,6.334)$.

3 PROBLEM-03

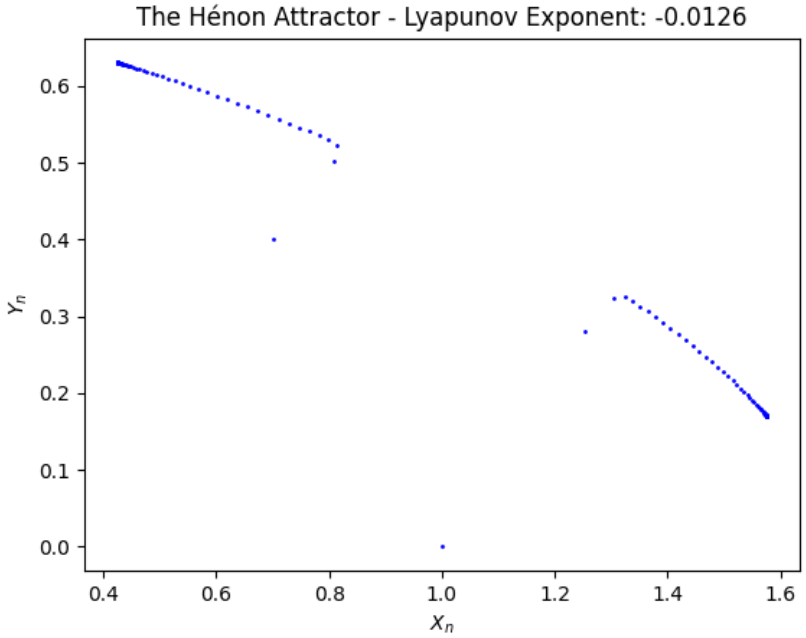
– $(a,b)=0.3$, $x_0=0$ and $x_0=0.00001$.

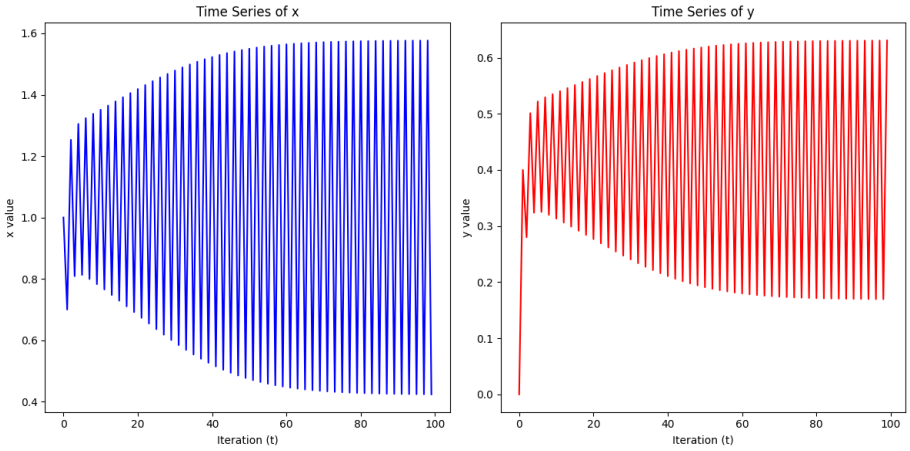
- $x_0=0$.




Time Series of x
x value
Iteration (t)
Time Series of y
y value
Iteration (t)

- $x_0 = 0.00001$.

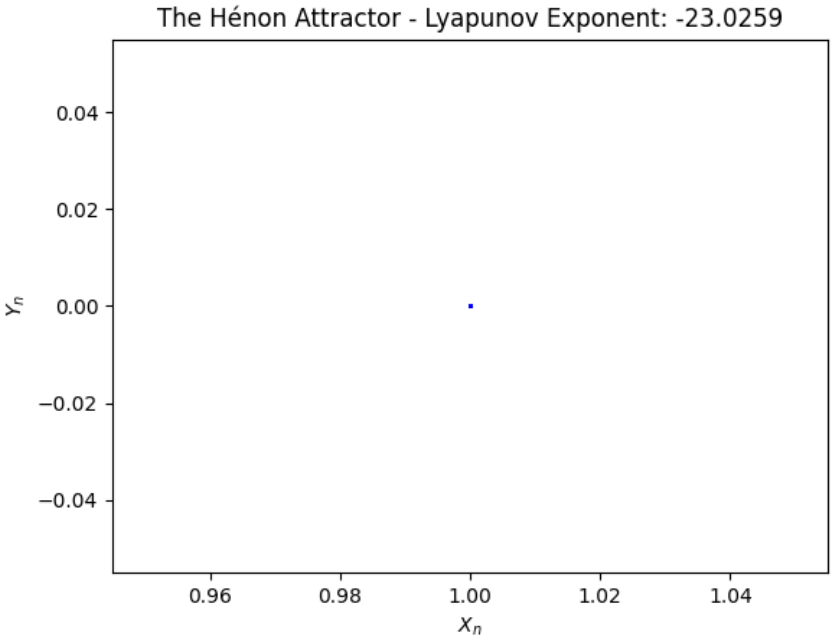

The Hénon Attractor - Lyapunov Exponent: -0.0126
Y_n
X_n

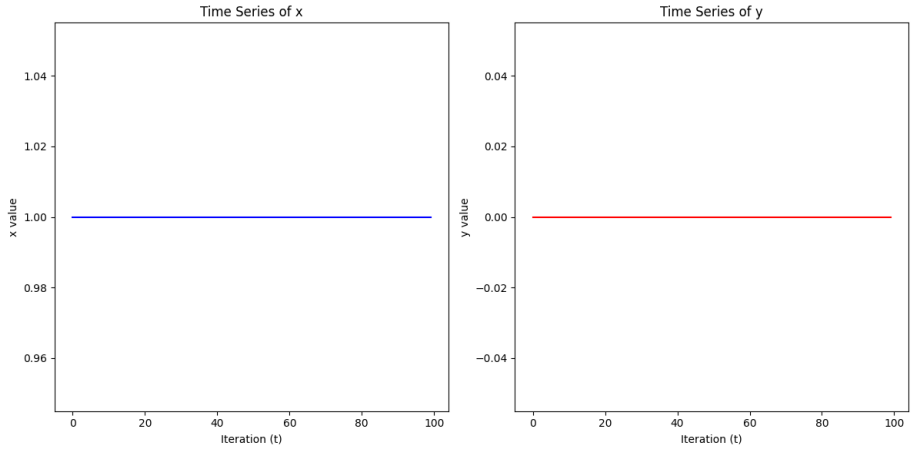


As we can observe from the Lyapunov exponent(It indicates that the system may be converging to a stable - periodic orbit) and from the time series plot we deduct that the 2 trajectories are periodic. We conclude that the the initial conditions don't play significant role when the sytem is periodic.

– Various values for a, $b = 0$, $x_0 = 0$ and $x_0 = 0$.

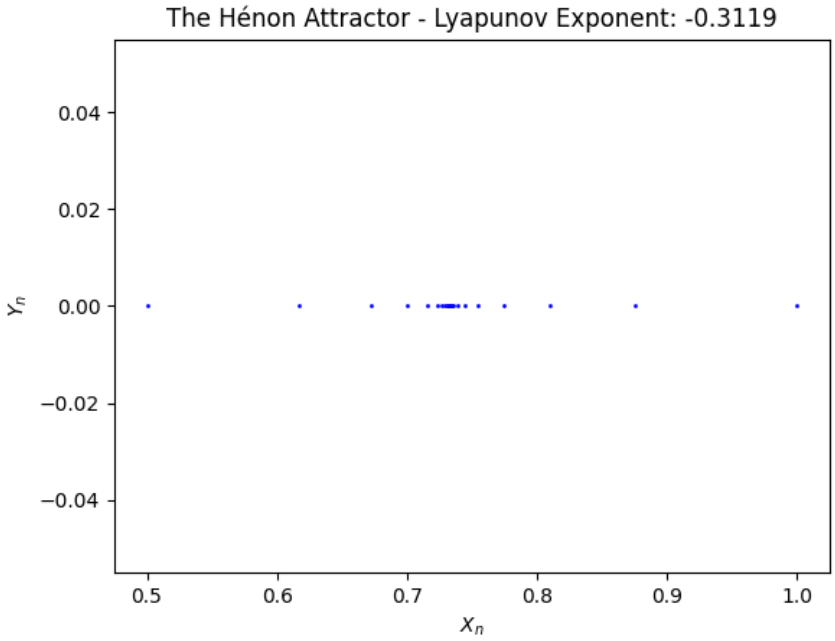
- $(a, b) = 0$.

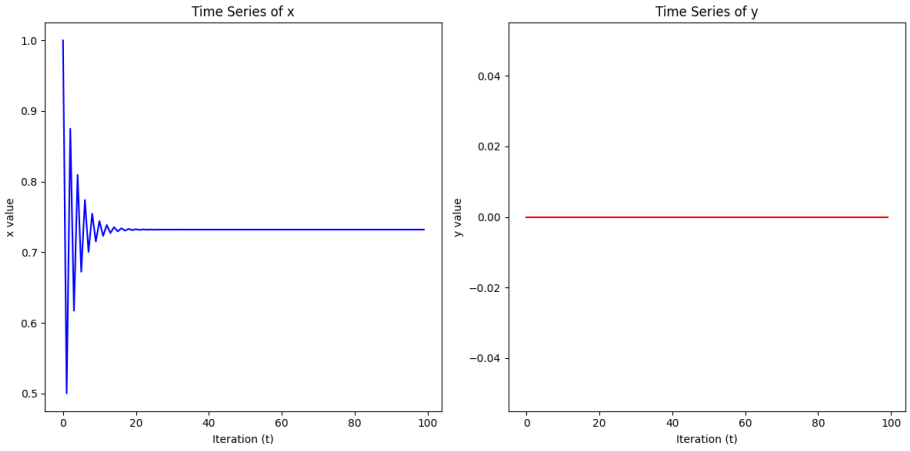




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

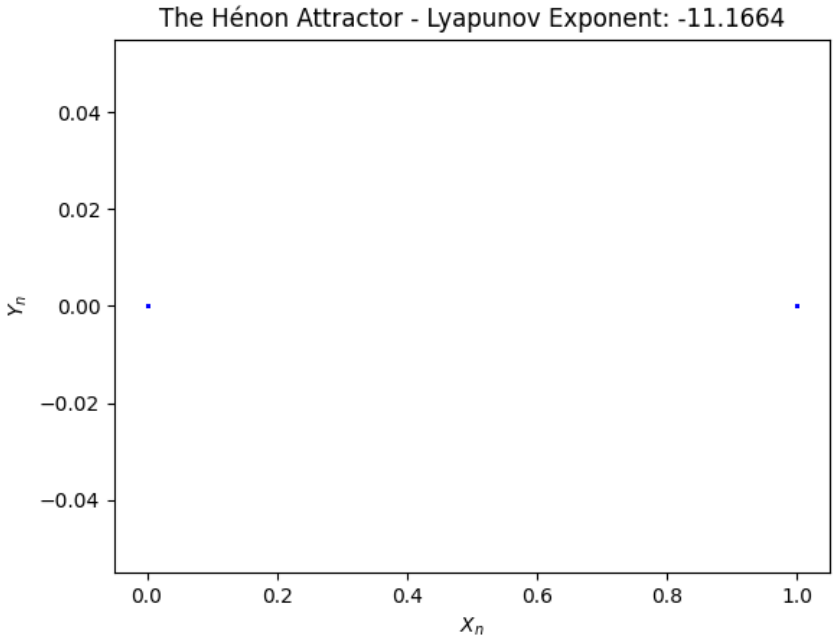
- $a = 0.5$ and $b = 0$.

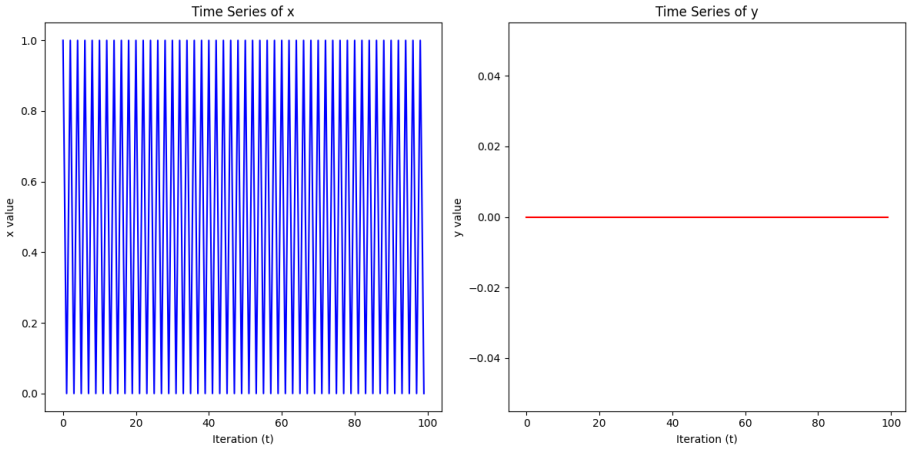




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

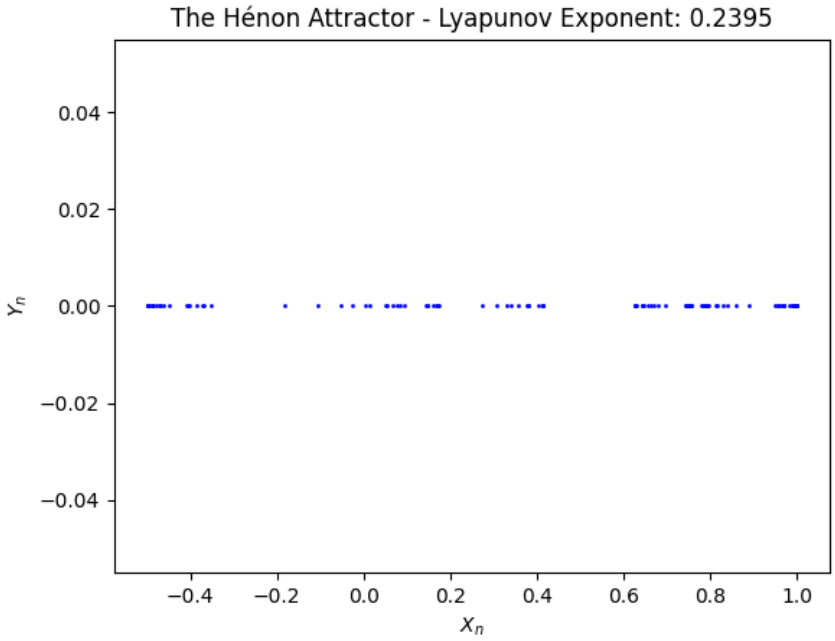
- $a = 1$ and $b = 0$.

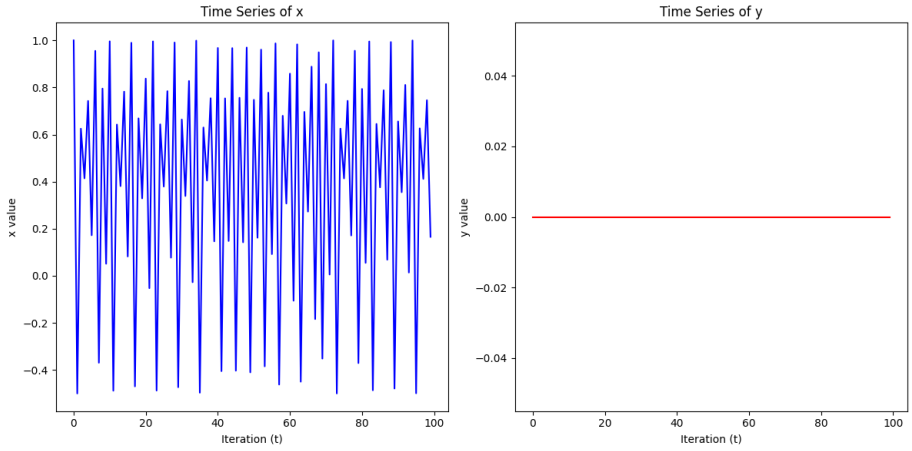




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

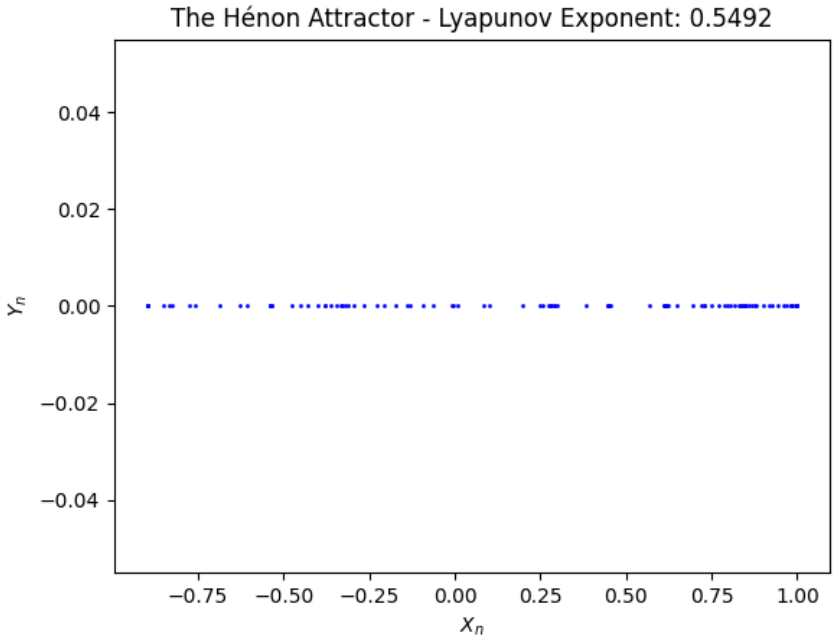
- $a = 1.5$ and $b = 0$.

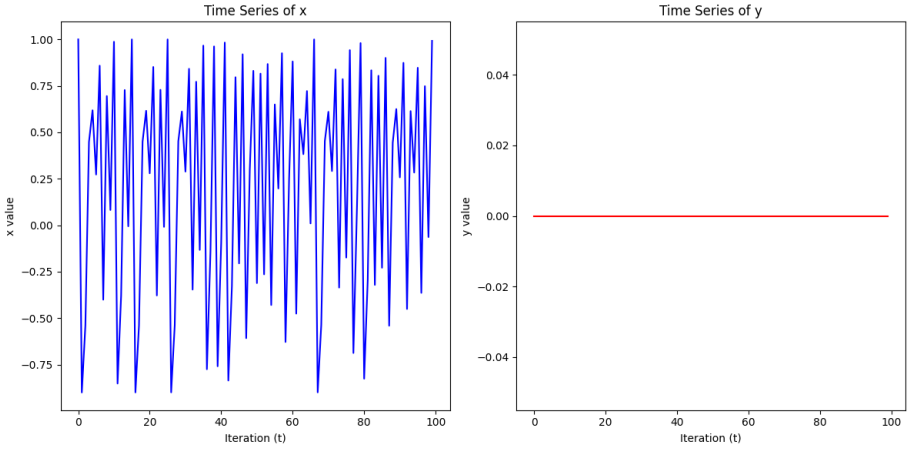




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

- $a = 2$ and $b = 0$.



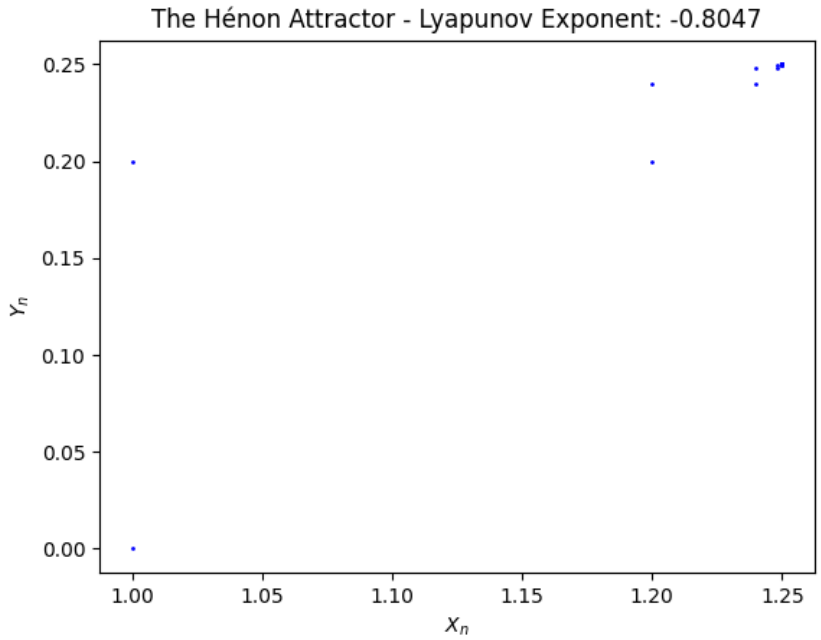


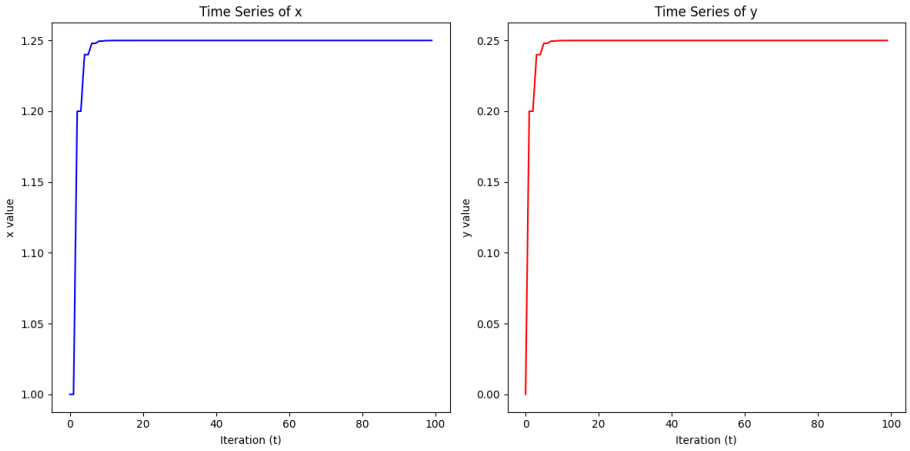
We observe chaos which is confirmed by the the Lyaponov exponent and the time series plots.

We conclude that when b = 0 the Hennon Attractor is bound to be zero in the y-Axis acting chaotic or becoming some fixed-point(s), according to the value that we give to a. Bigger a values transition the system from periodic to chaotic.

– Various values for x_0 and b and a between 0 and 0.32.

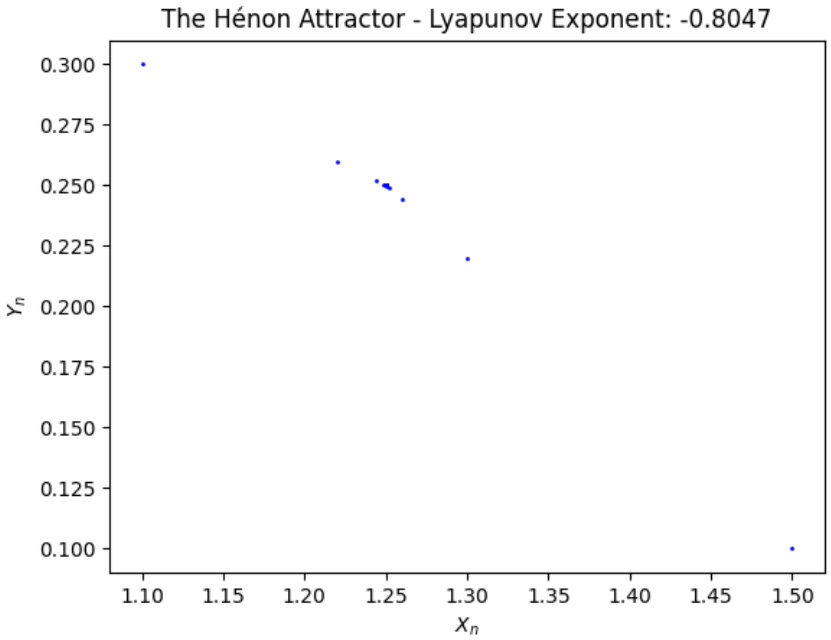
- $(a, b) = (0, 0.2)$, $x_0 = 0$ and $x_0 = 0$.

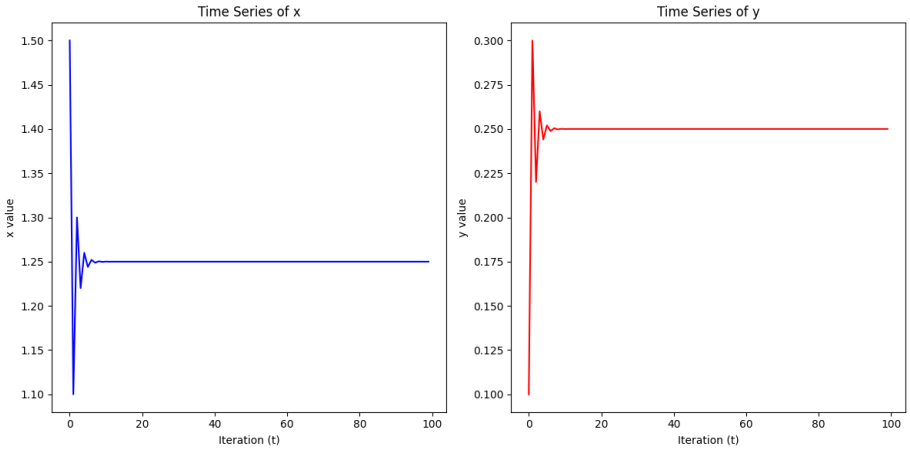




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

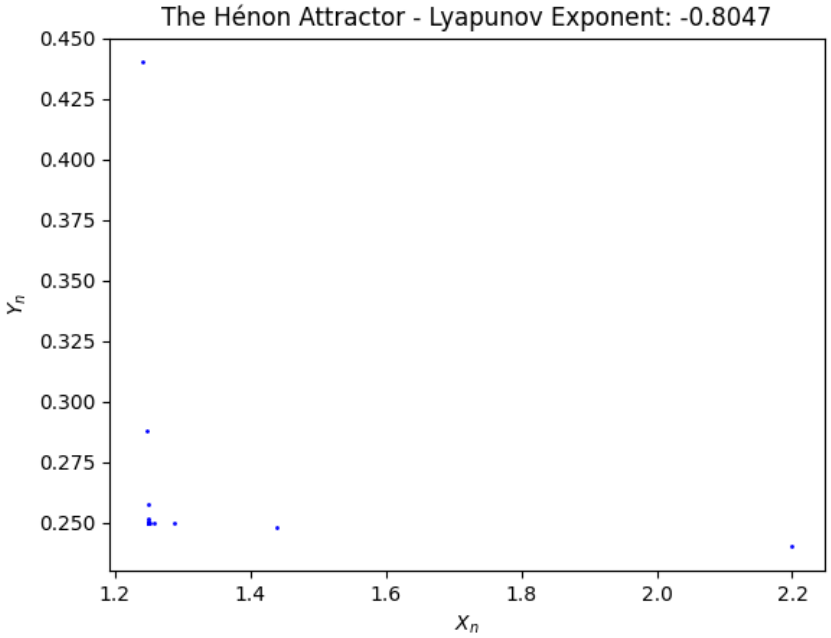
- $(a, b) = (0, 0.2)$, $x_0 = 0.5$ and $x_0 = 0.5$.

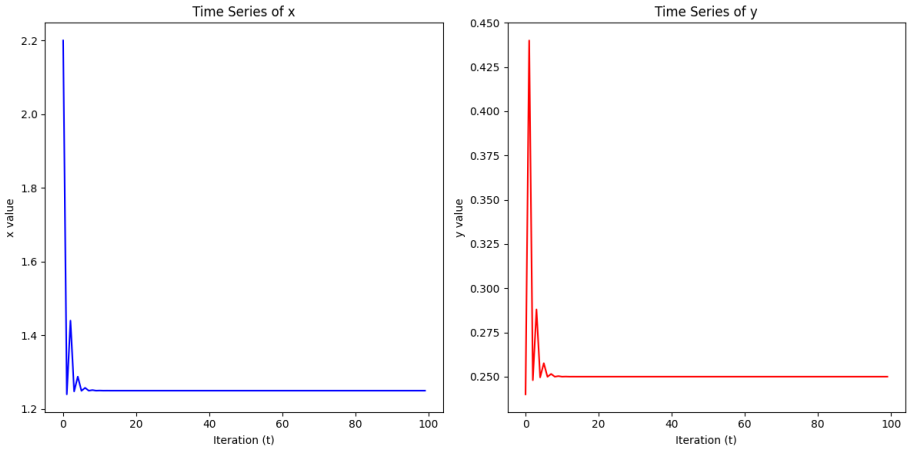




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

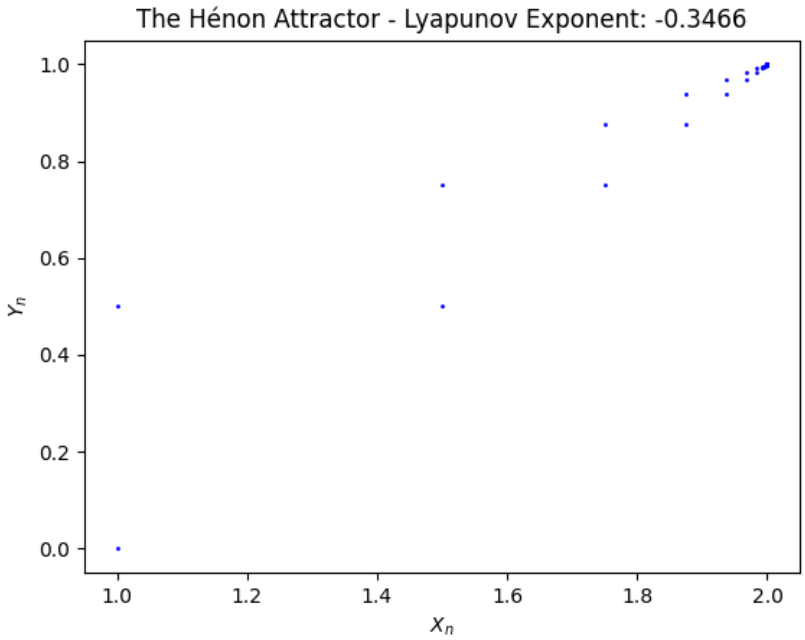
- $(a, b) = (0, 0.2)$, $x_0 = 1.2$ and $x_0 = 1.2$.

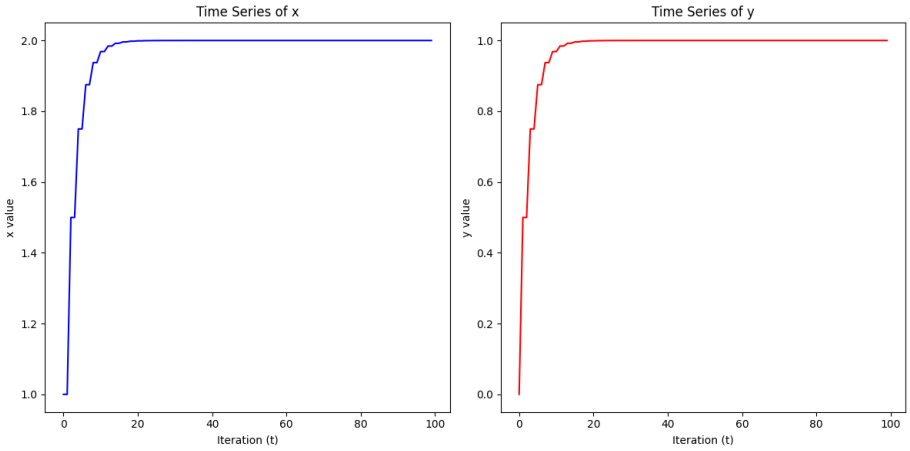




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

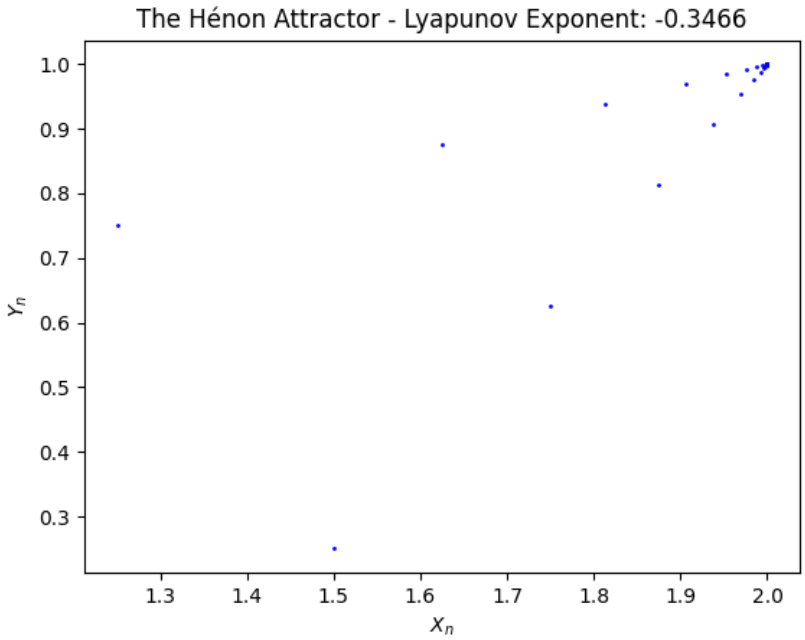
- $(a, b) = (0, 0.5)$, $x_0 = 0$ and $x_0 = 0$.

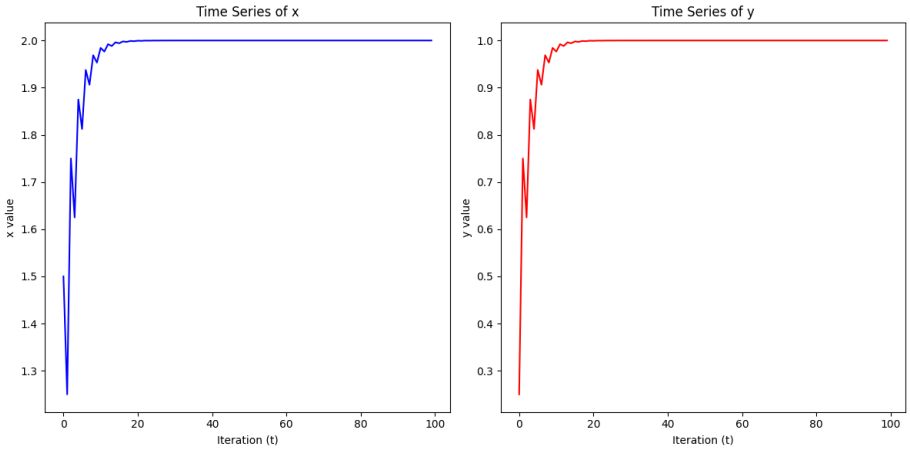




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

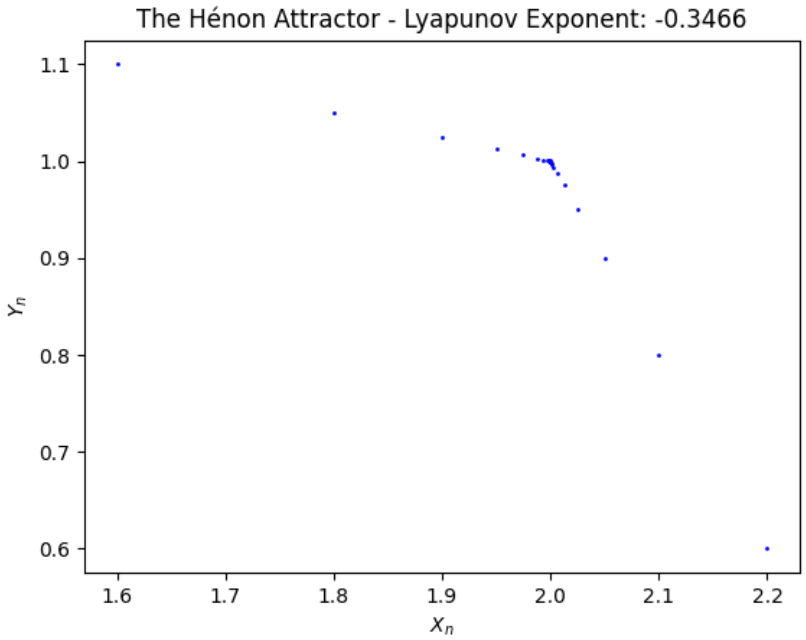
- $(a, b) = (0, 0.5)$, $x_0 = 0.5$ and $x_0 = 0.5$.

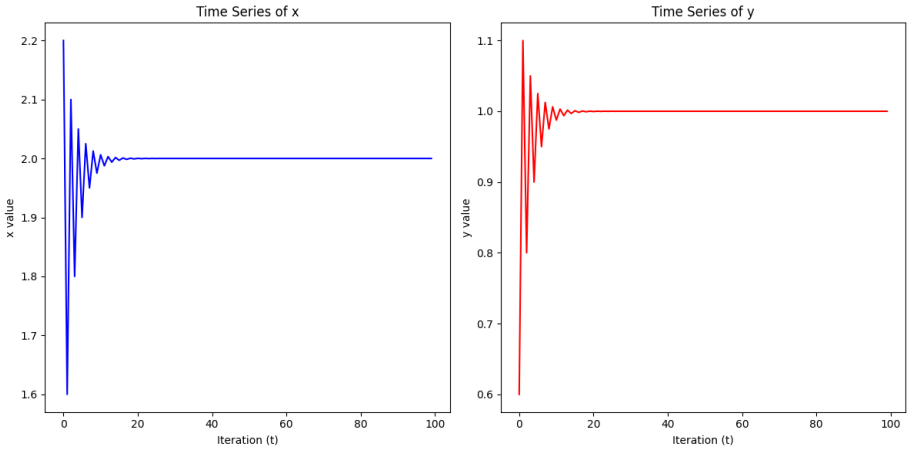




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

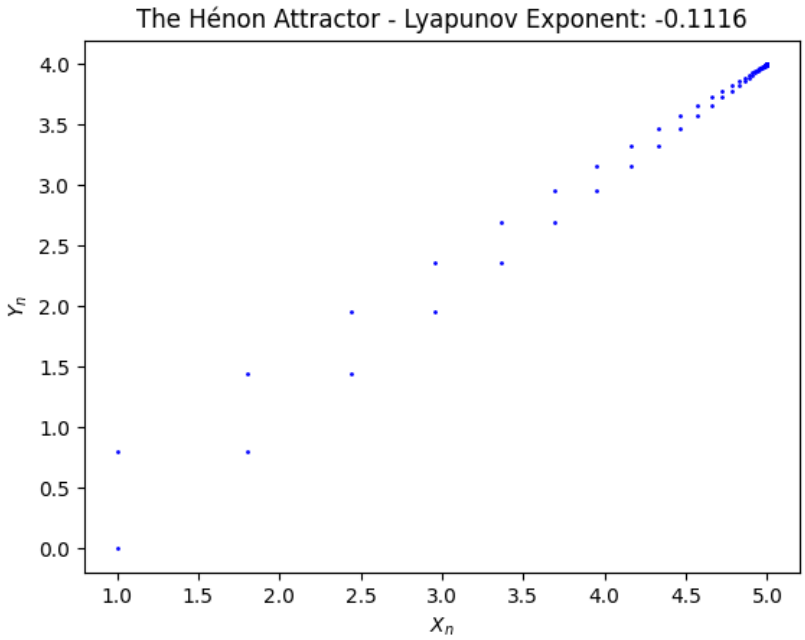
- $(a, b) = (0, 0.5)$, $x_0 = 1.2$ and $x_0 = 1.2$.

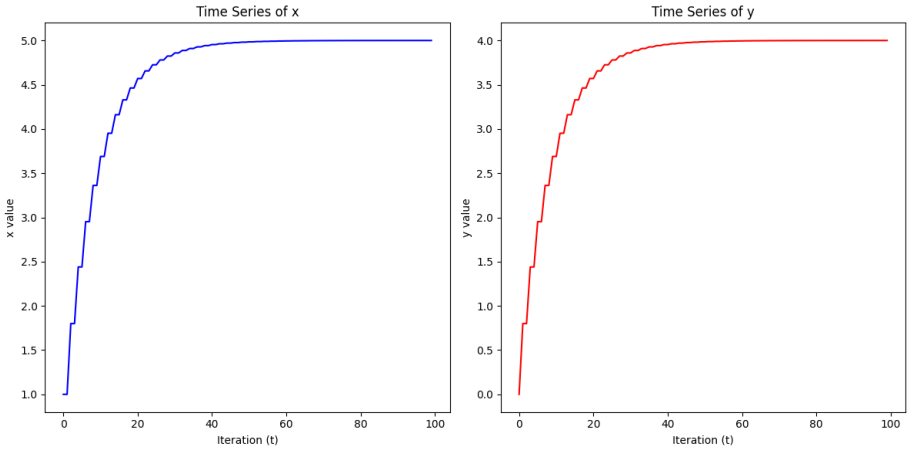




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

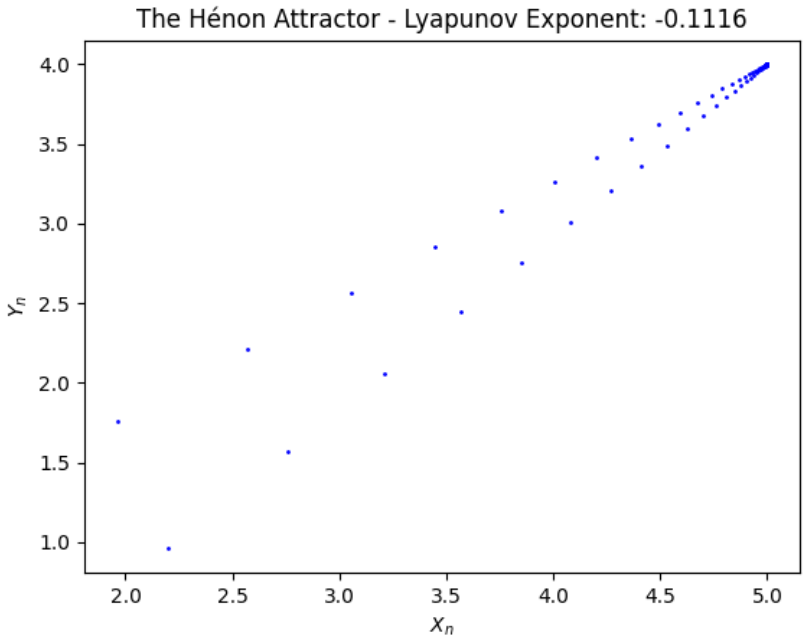
- $(a, b) = (0, 0.8)$, $x_0 = 0$ and $x_0 = 0$.

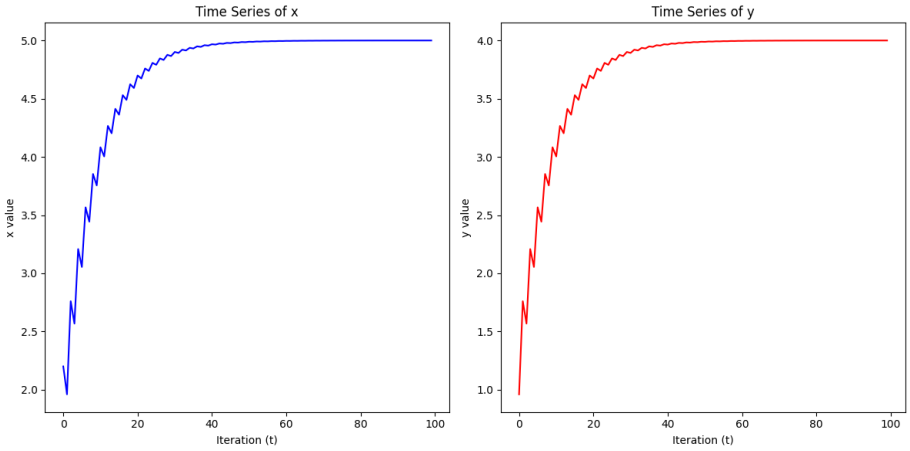




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

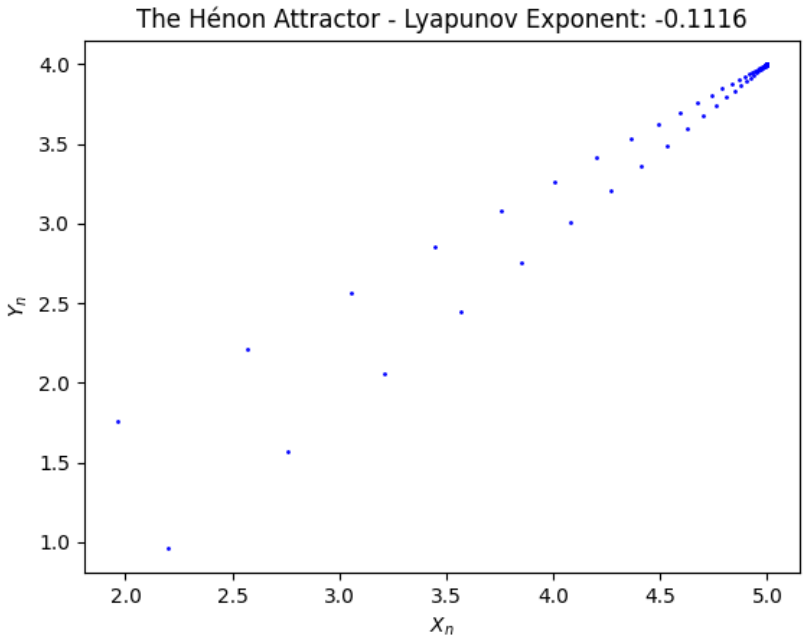
- $(a, b) = (0, 0.8)$, $x_0 = 0.5$ and $x_0 = 0.5$.

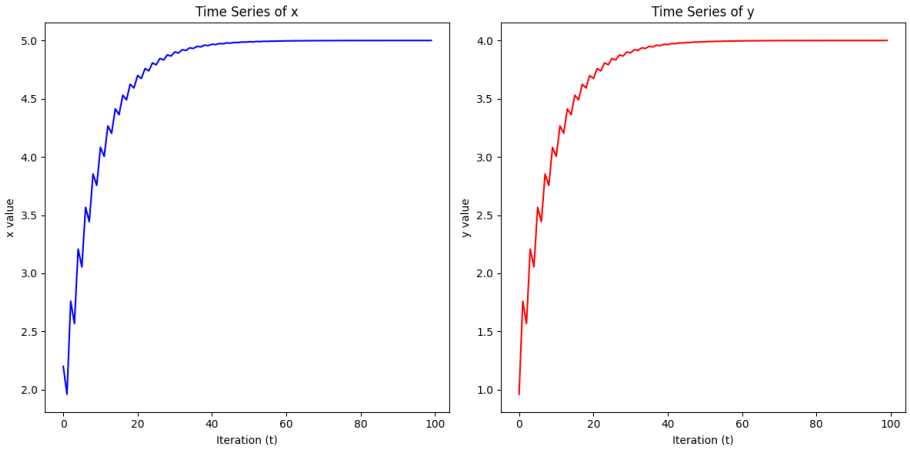




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

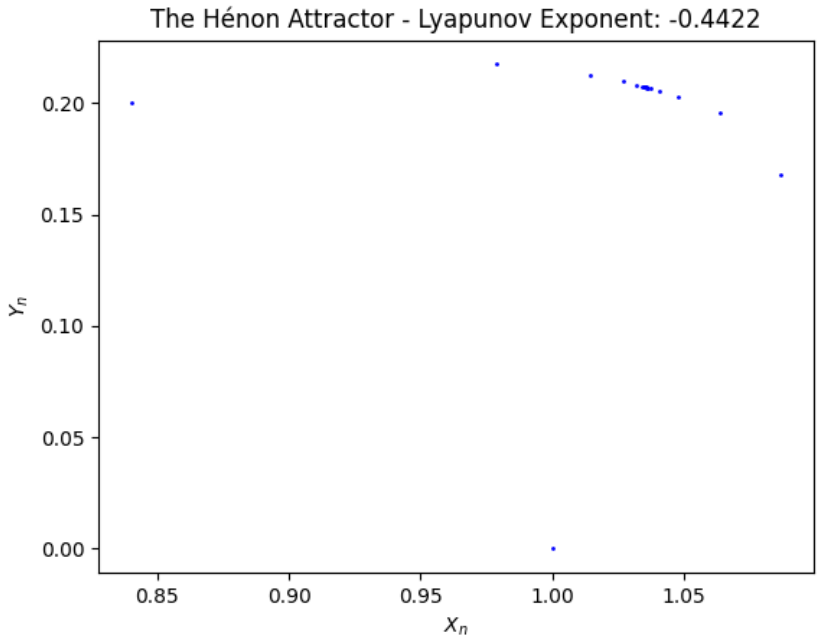
- $(a, b) = (0, 0.8)$, $x_0 = 1.2$ and $x_0 = 1.2$.

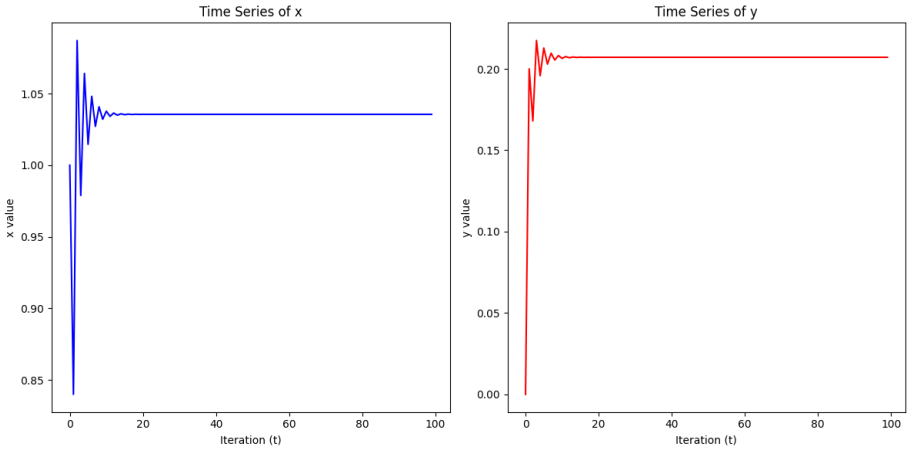




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

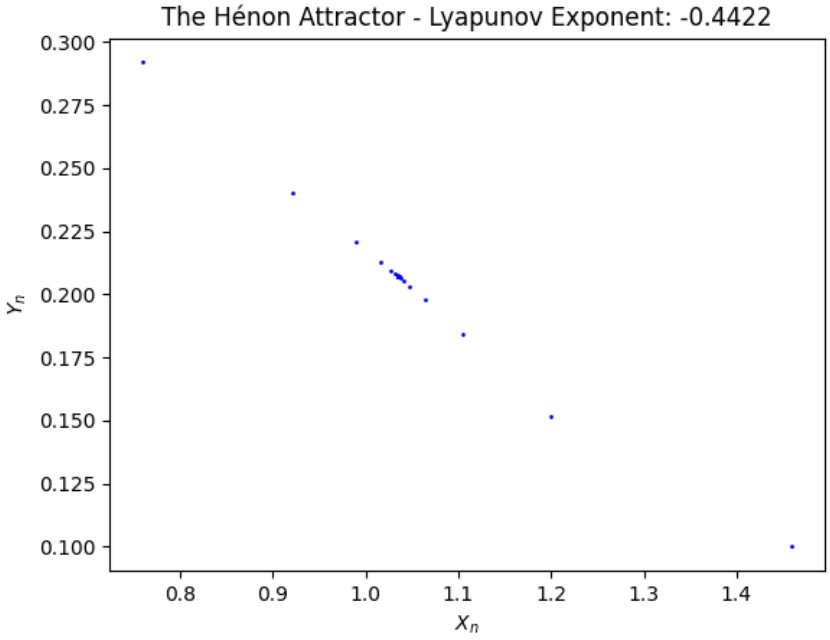
- $(a, b) = (0.16, 0.2)$, $x_0 = 0$ and $x_0 = 0$.

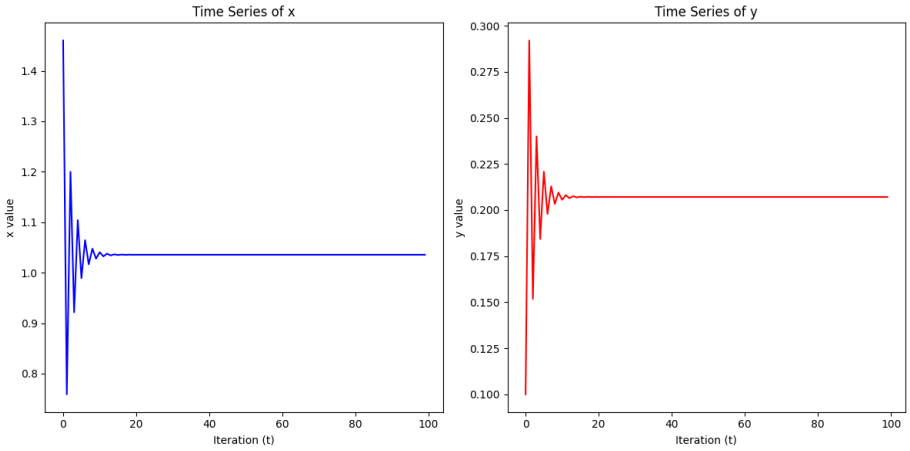




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

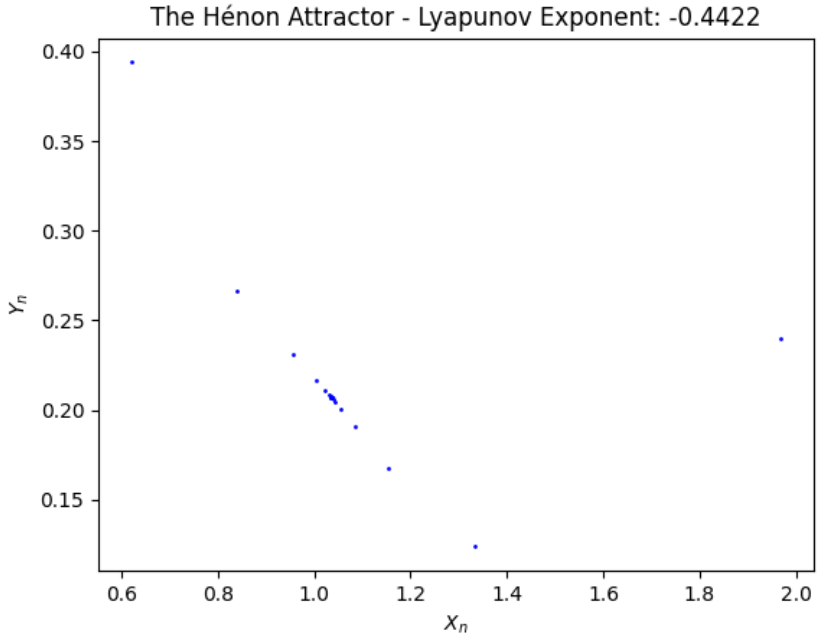
- $(a, b) = (0.16, 0.2)$, $x_0 = 0.5$ and $x_0 = 0.5$.

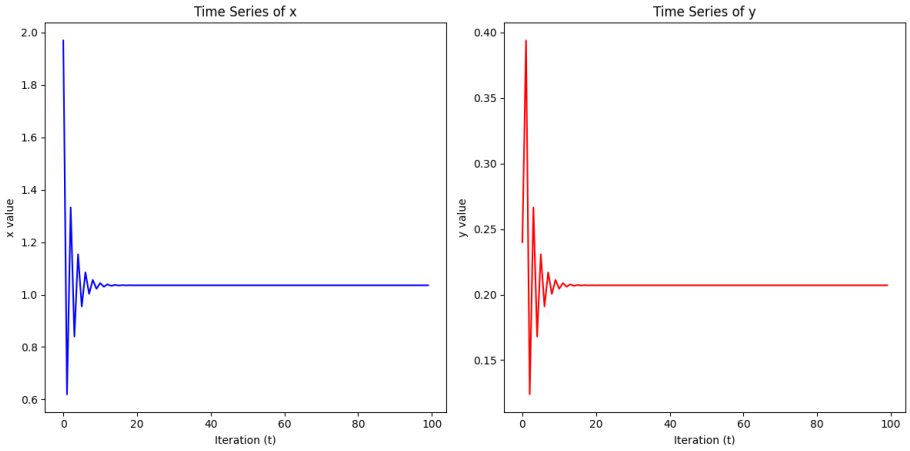




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

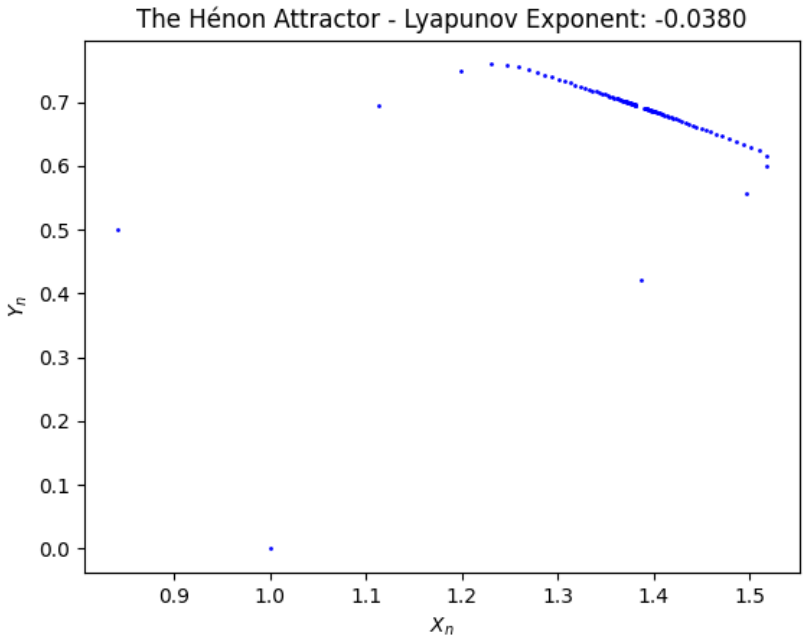
- $(a, b) = (0.16, 0.2)$, $x_0 = 1.2$ and $x_0 = 1.2$.

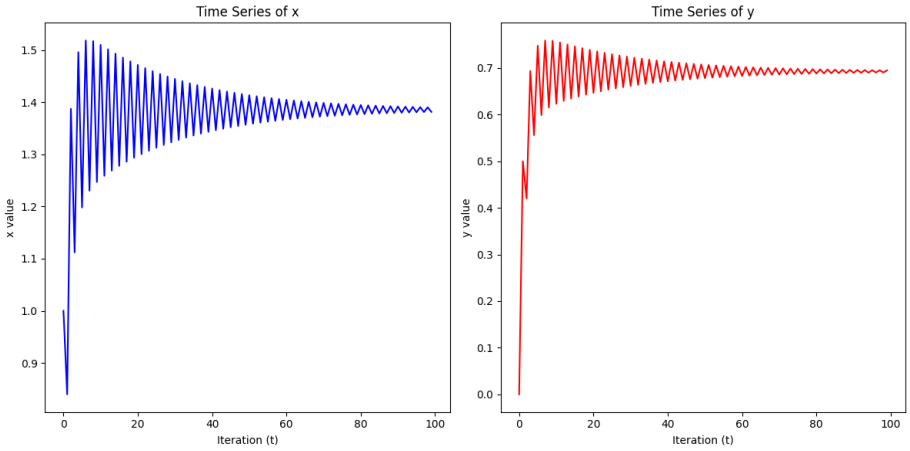




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

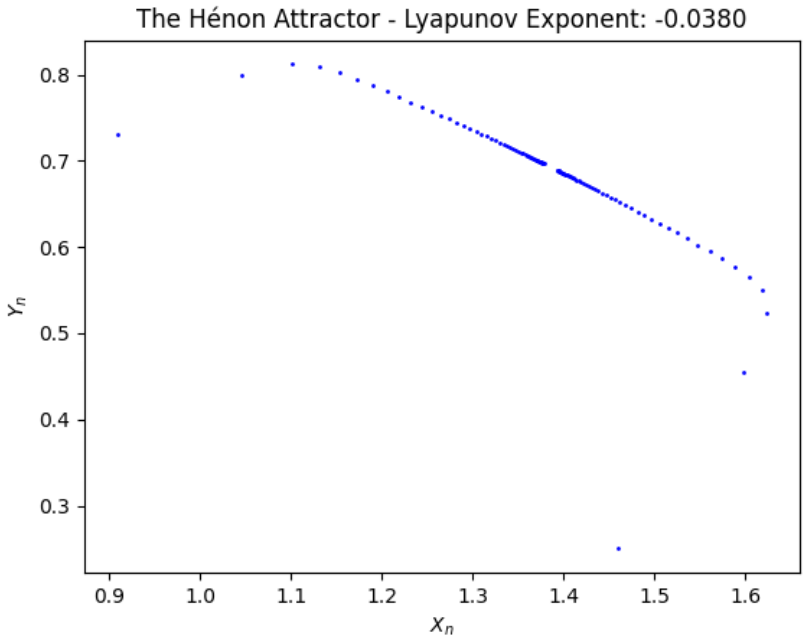
- $(a, b) = (0.16, 0.5)$, $x_0 = 0$ and $x_0 = 0$.

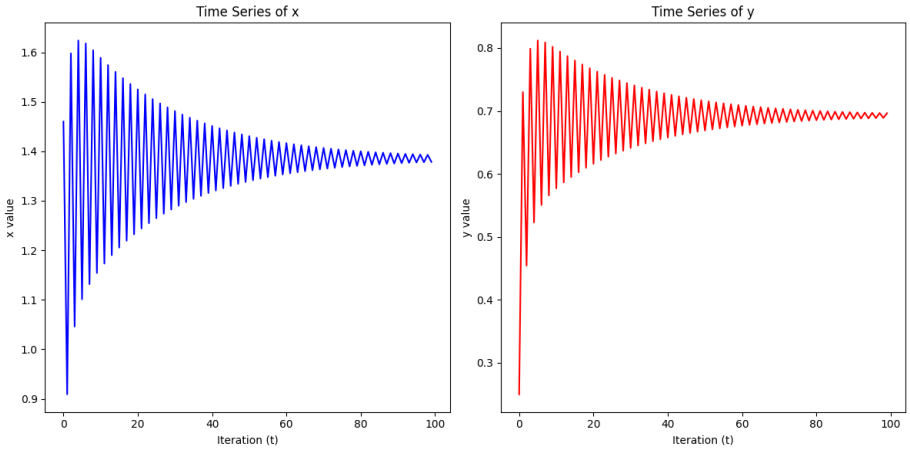




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

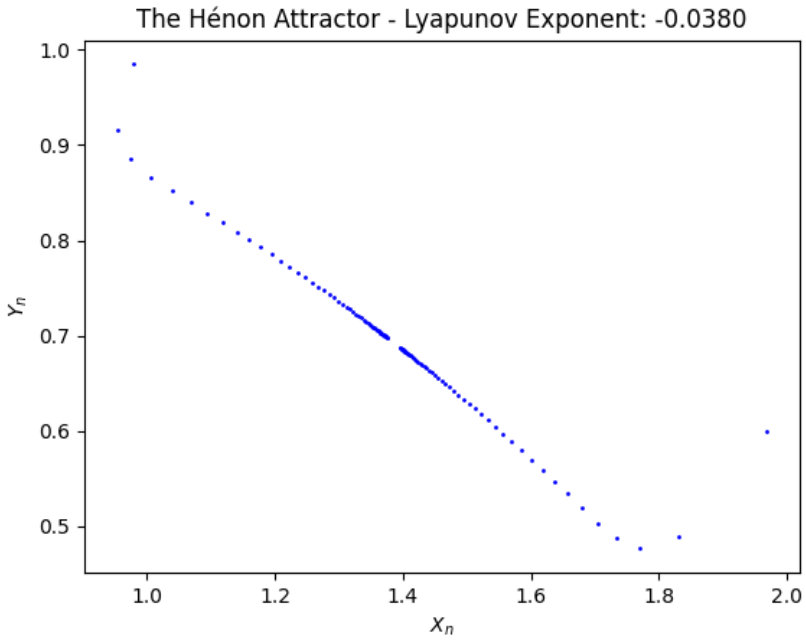
- $(a, b) = (0.16, 0.5)$, $x_0 = 0.5$ and $x_0 = 0.5$.

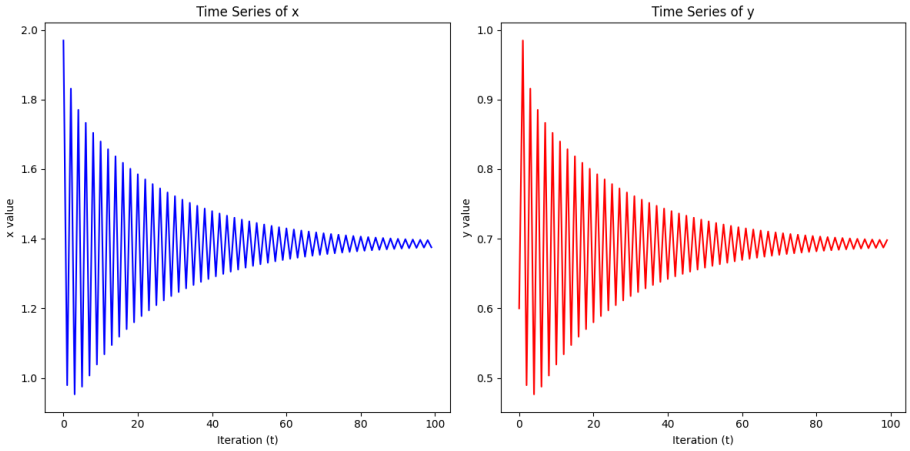




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

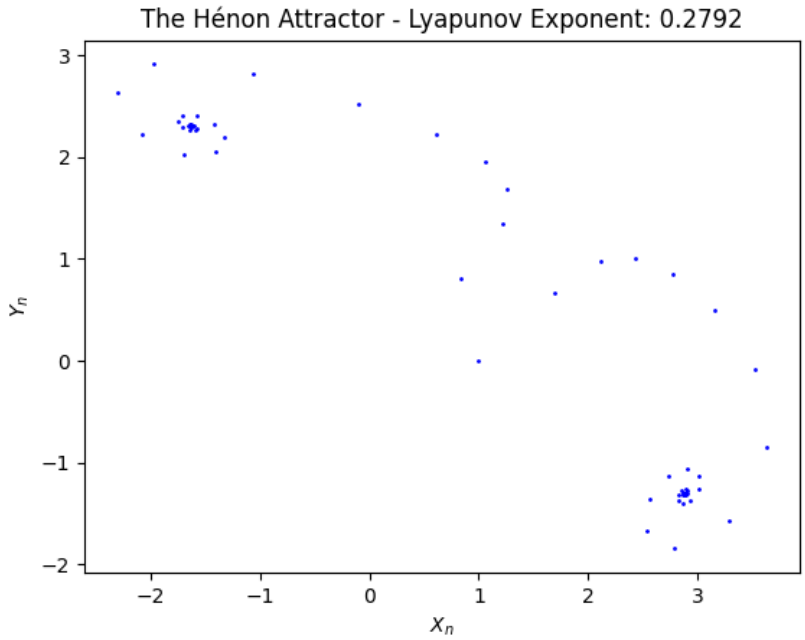
- $(a, b) = (0.16, 0.5)$, $x_0 = 1.2$ and $x_0 = 1.2$.

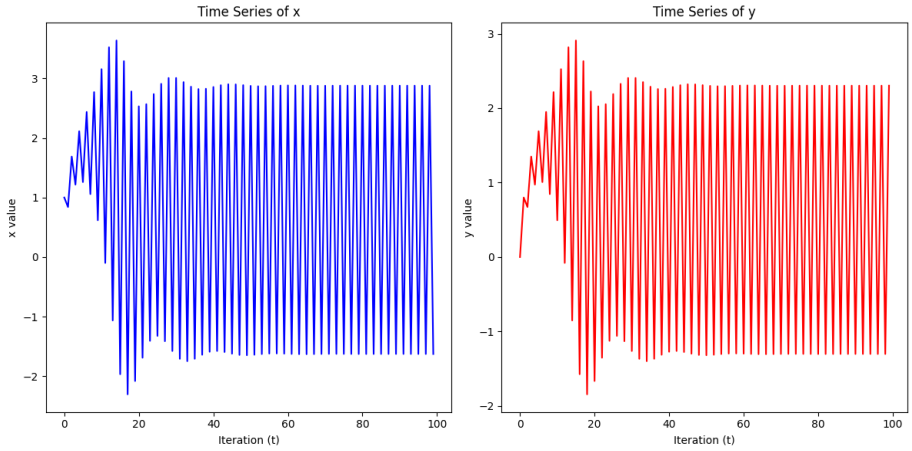




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

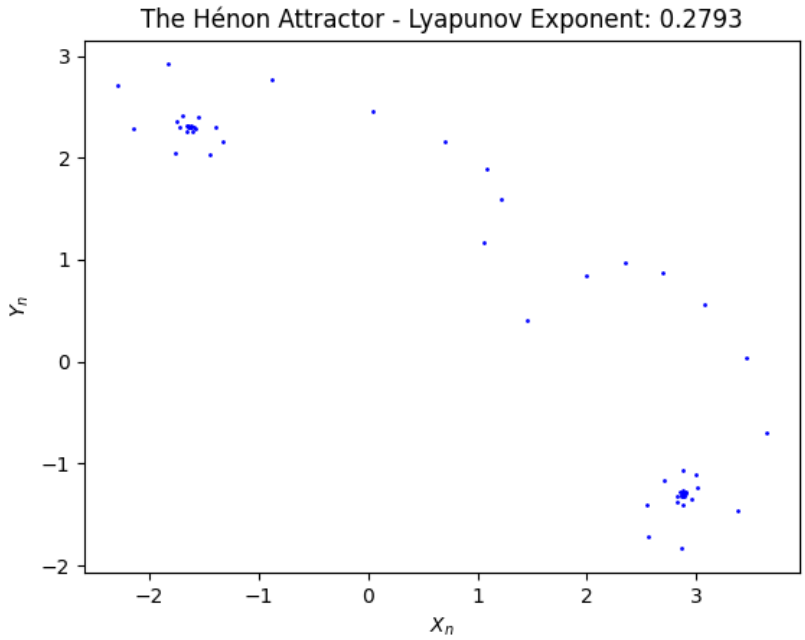
- $(a, b) = (0.16, 0.8)$, $x_0 = 0$ and $x_0 = 0$.

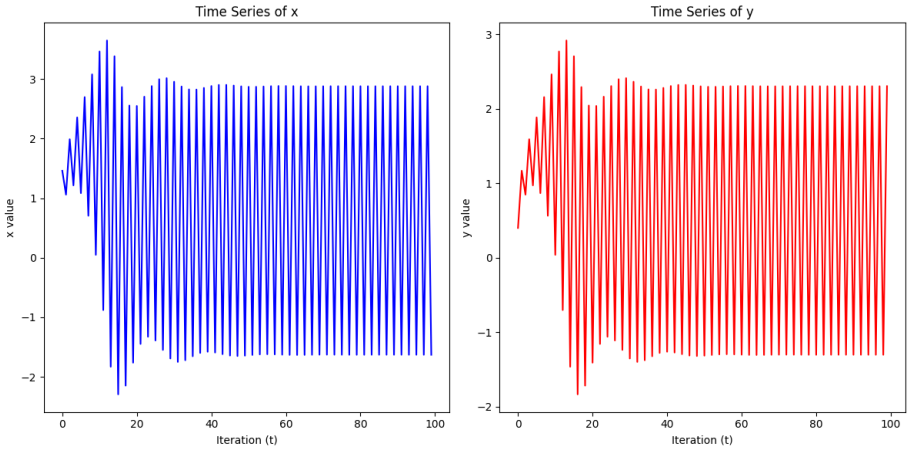




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

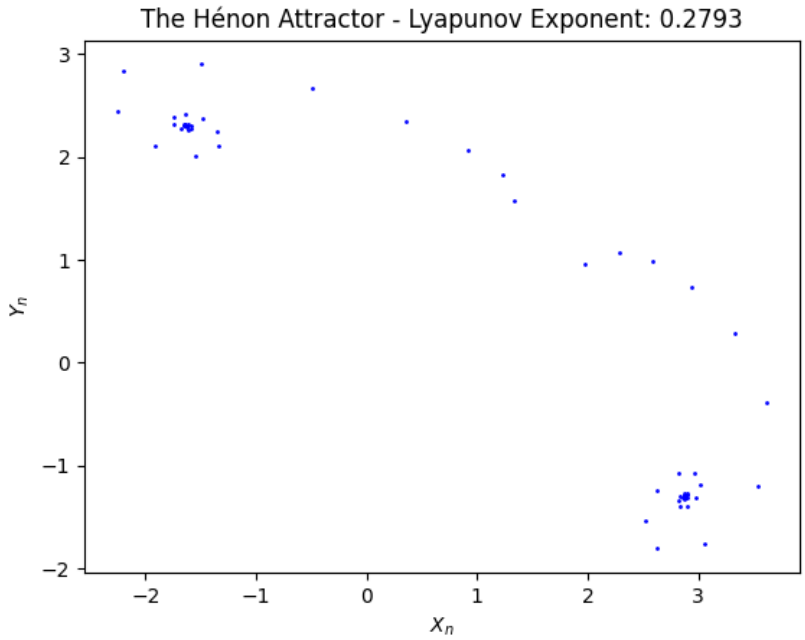
- $(a, b) = (0.16, 0.8)$, $x_0 = 0.5$ and $x_0 = 0.5$.

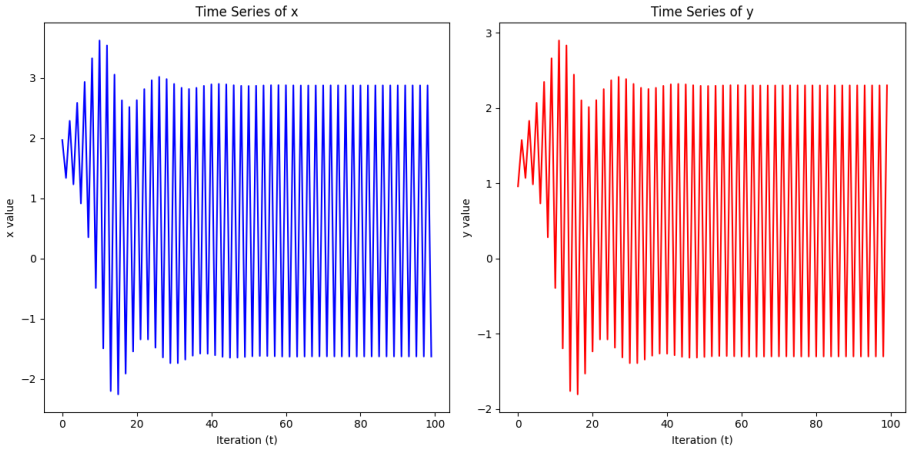




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

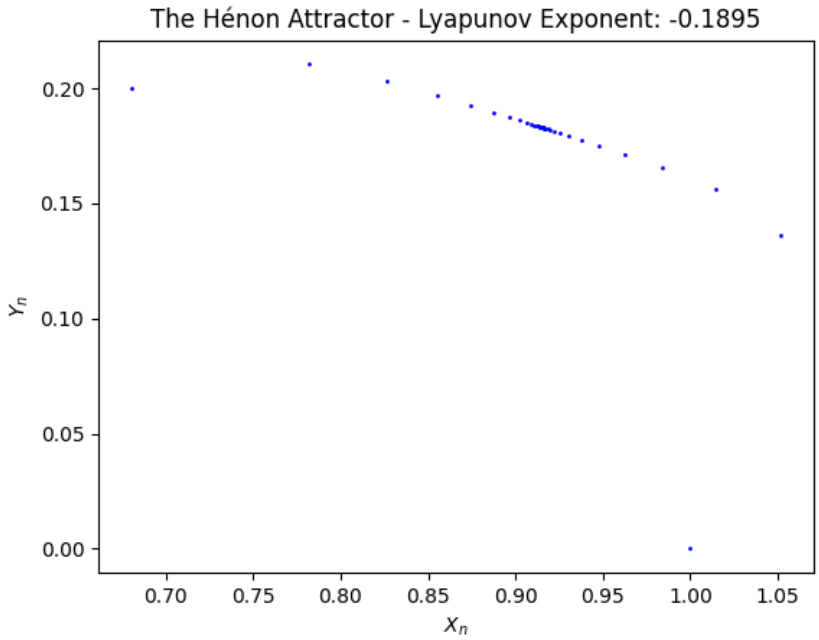
- $(a, b) = (0.16, 0.8)$, $x_0 = 1.2$ and $x_0 = 1.2$.

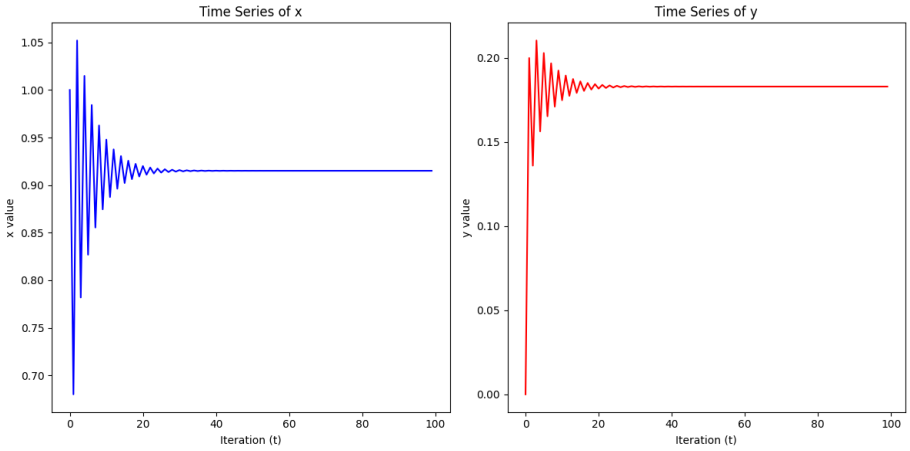




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

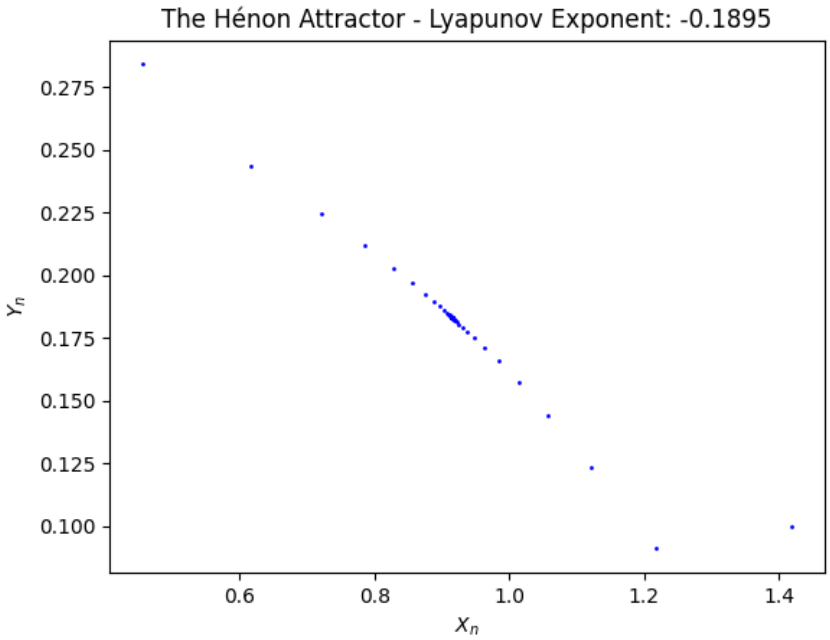
- $(a, b) = (0.32, 0.2)$, $x_0 = 0$ and $x_0 = 0$.

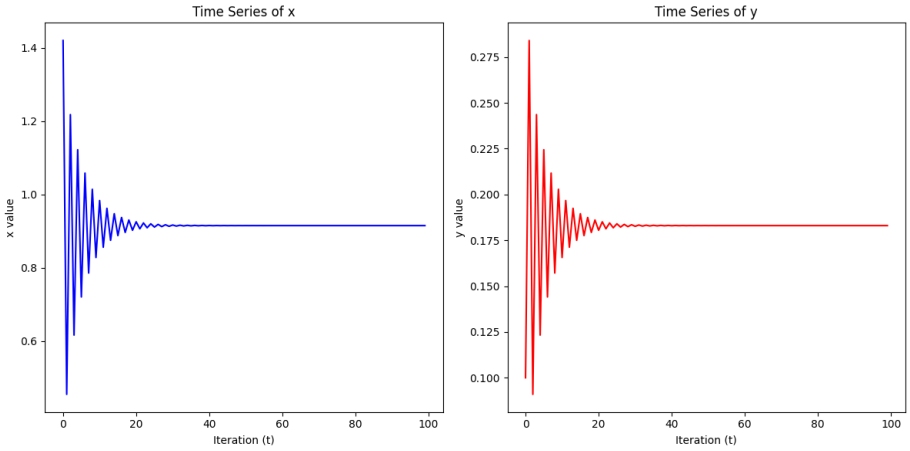




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

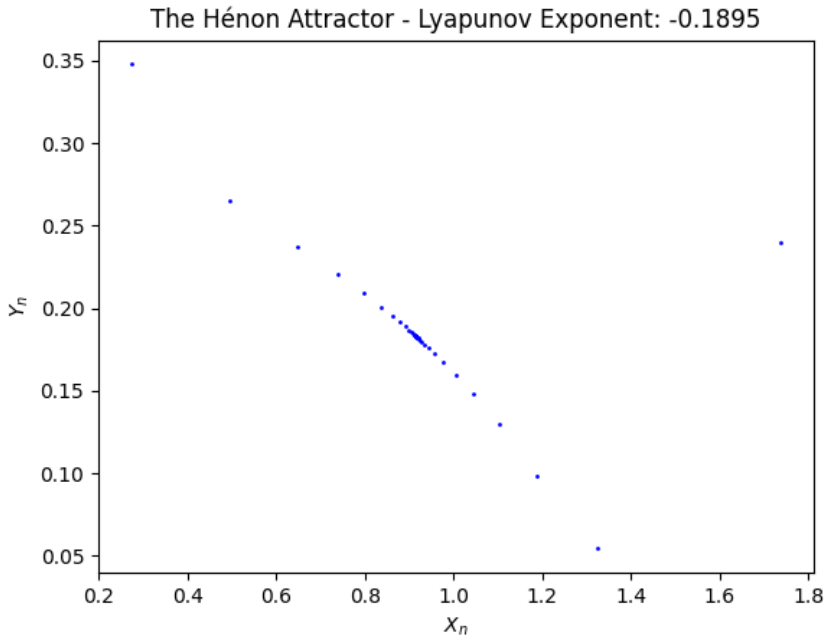
- $(a, b) = (0.32, 0.2)$, $x_0 = 0.5$ and $x_0 = 0.5$.

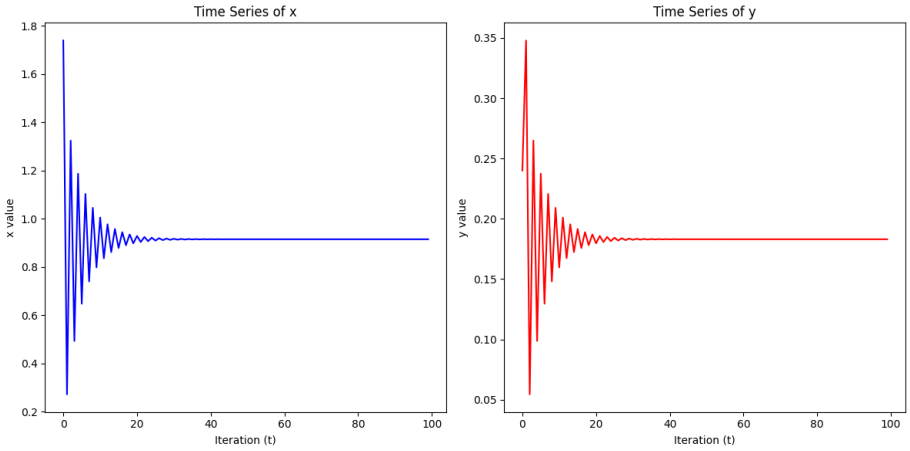




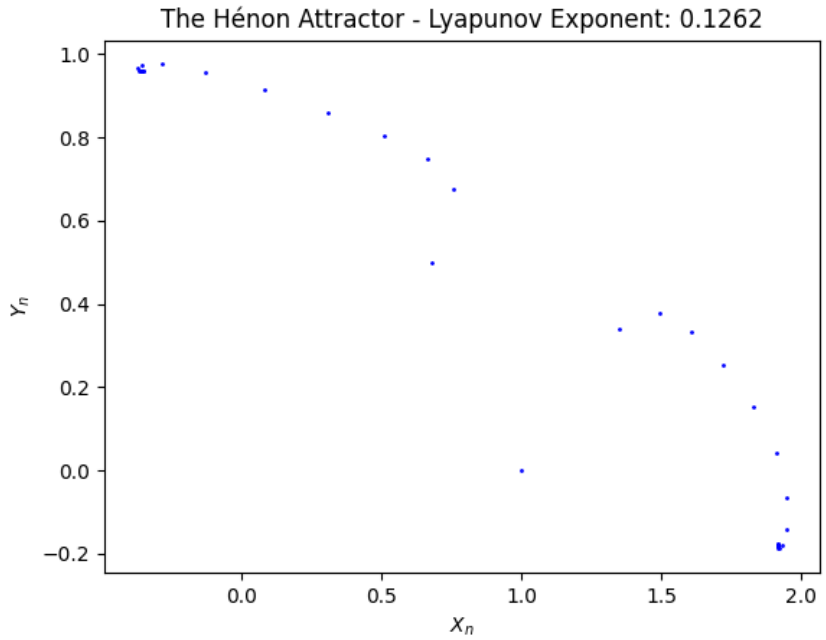
We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

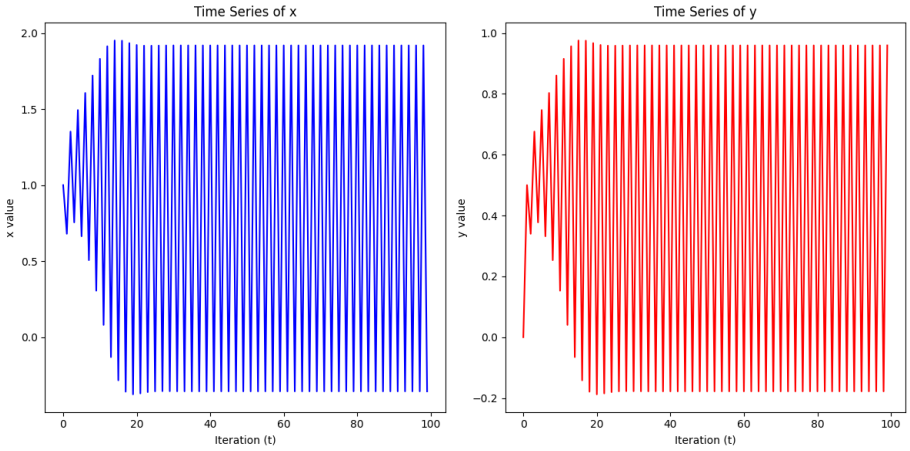
- $(a, b) = (0.32, 0.2)$, $x_0 = 1.2$ and $x_0 = 1.2$.





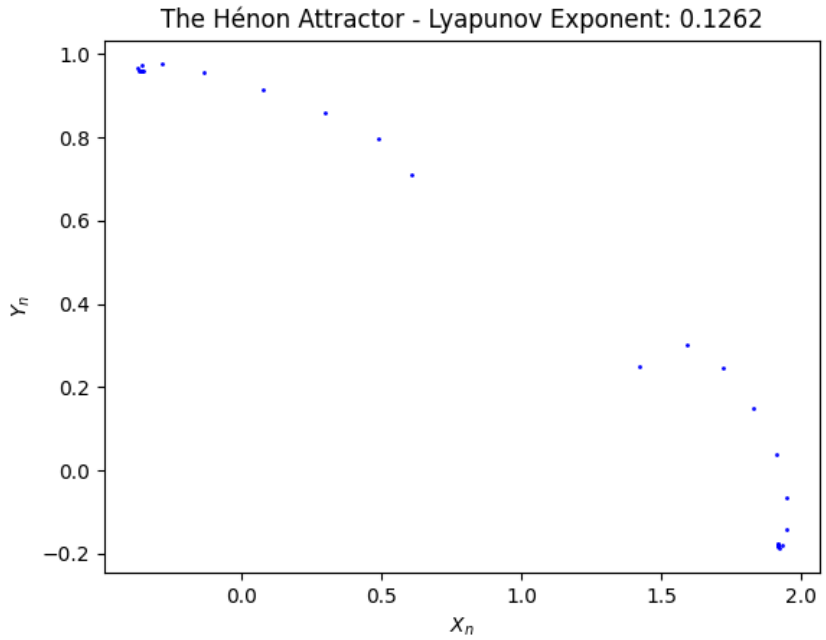
- $(a, b) = (0.32, 0.5)$, $x_0 = 0$ and $x_0 = 0$.

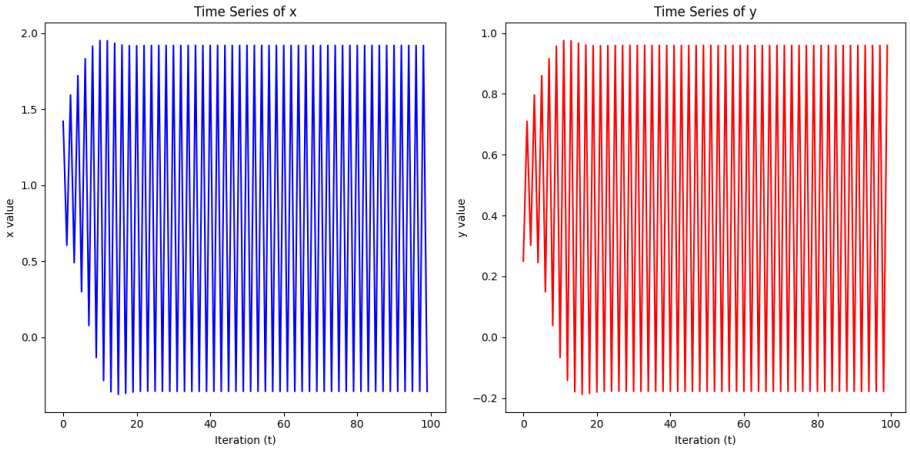




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

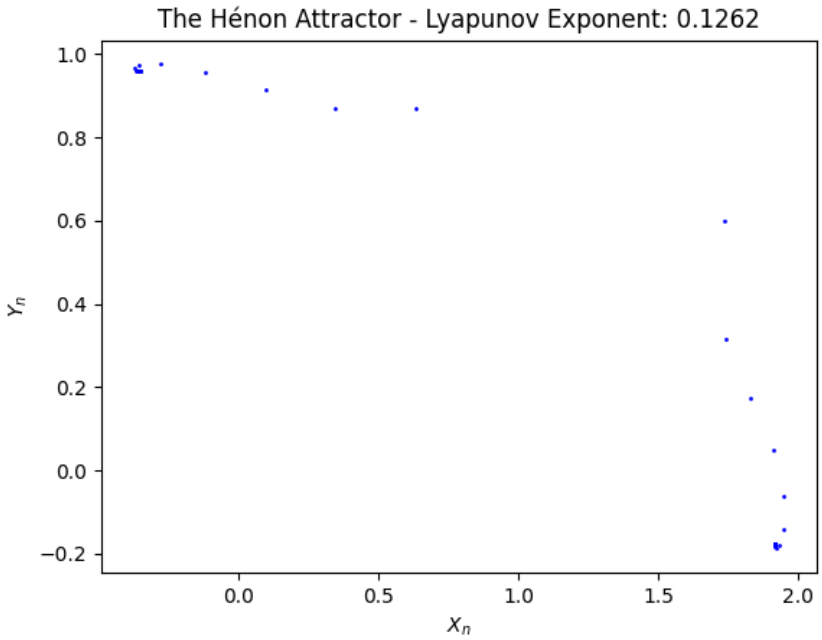
- $(a, b) = (0.32, 0.5)$, $x_0 = 0.5$ and $x_0 = 0.5$.

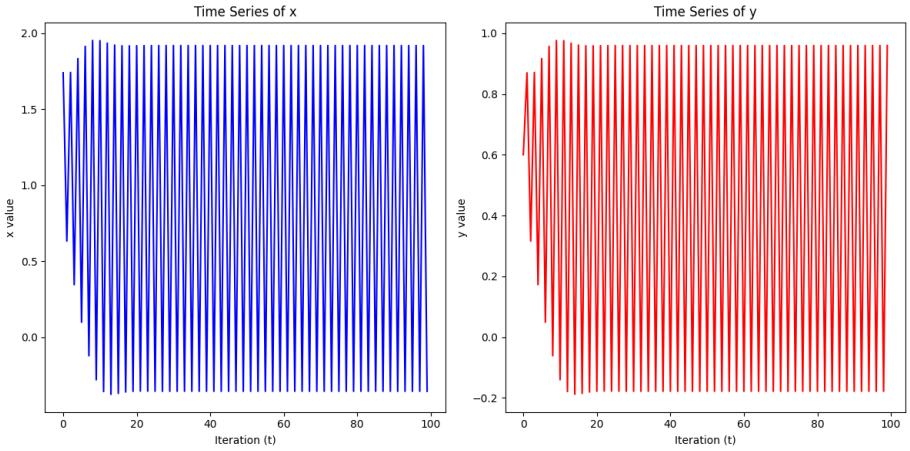




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

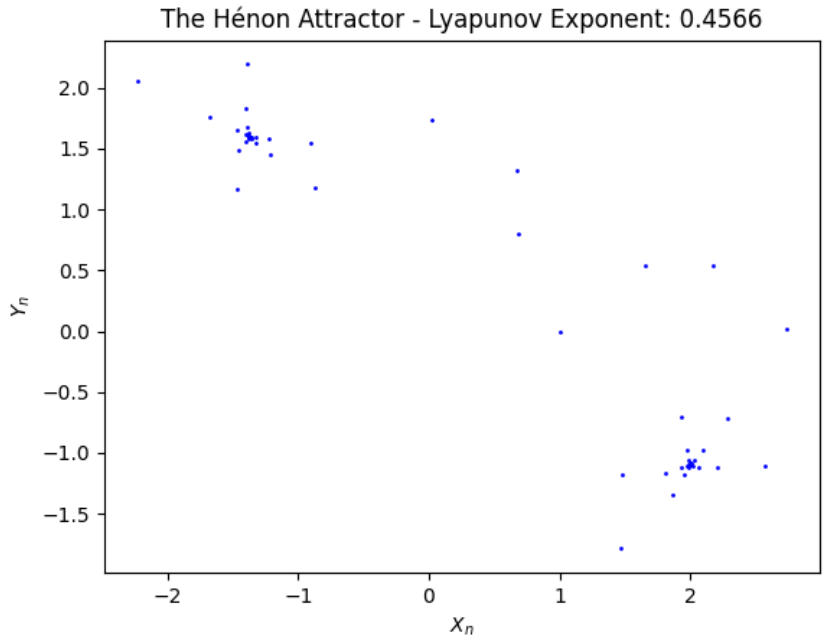
- $(a, b) = (0.32, 0.5)$, $x_0 = 1.2$ and $x_0 = 1.2$.

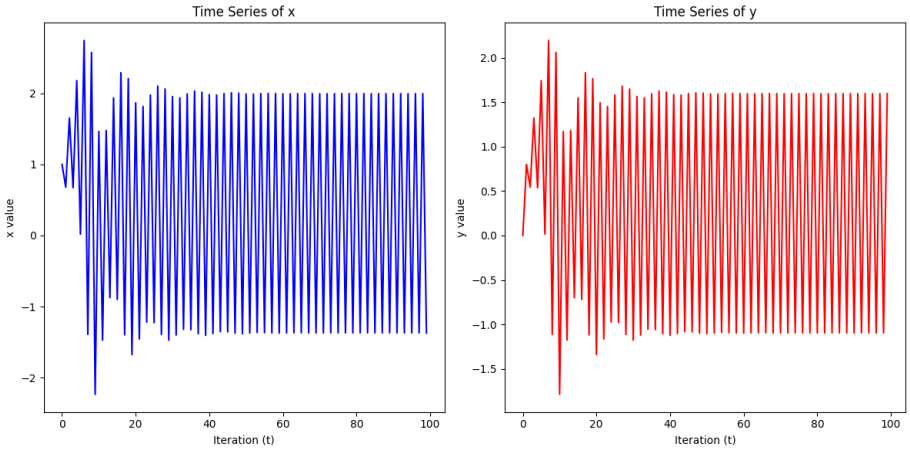




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

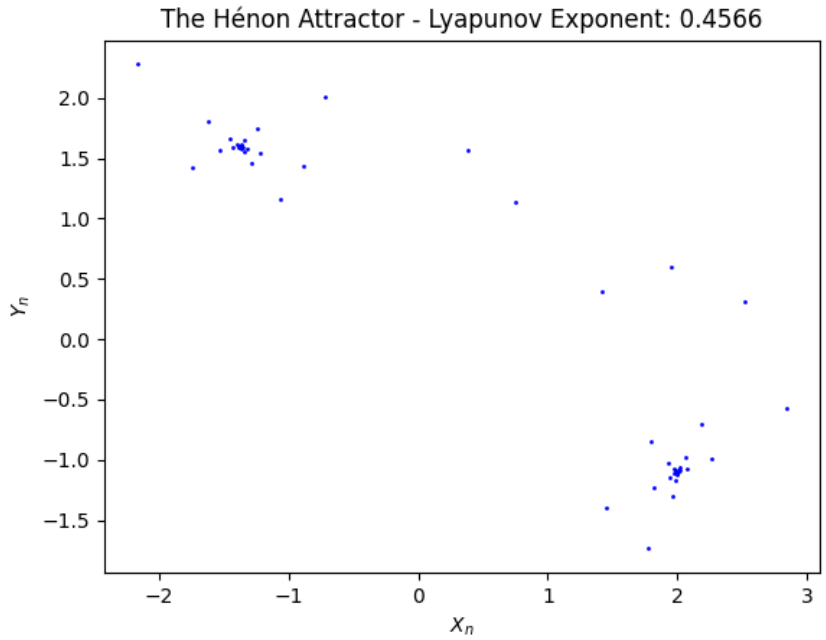
- $(a, b) = (0.32, 0.8)$, $x_0 = 0$ and $x_0 = 0$.

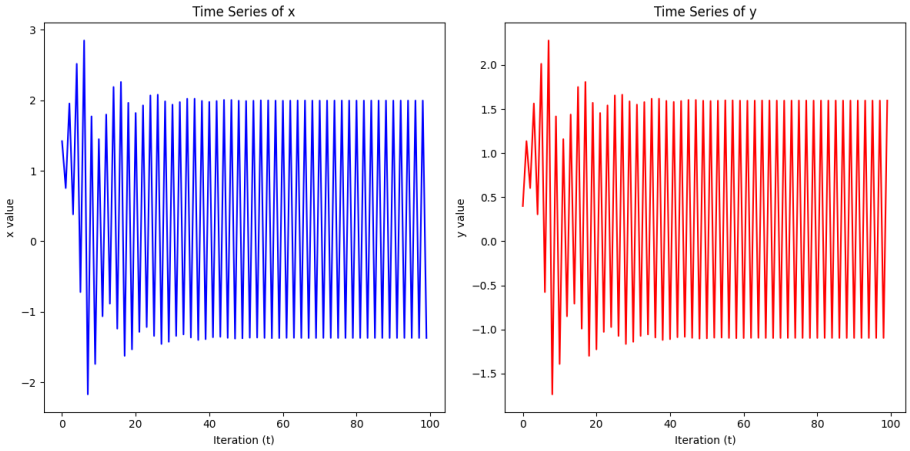




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

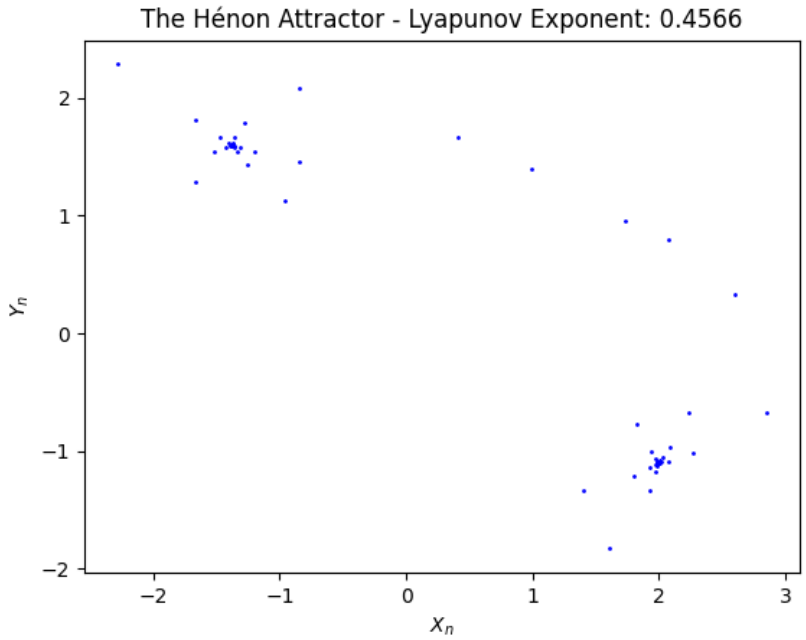
- $(a, b) = (0.32, 0.8)$, $x_0 = 0.5$ and $x_0 = 0.5$.

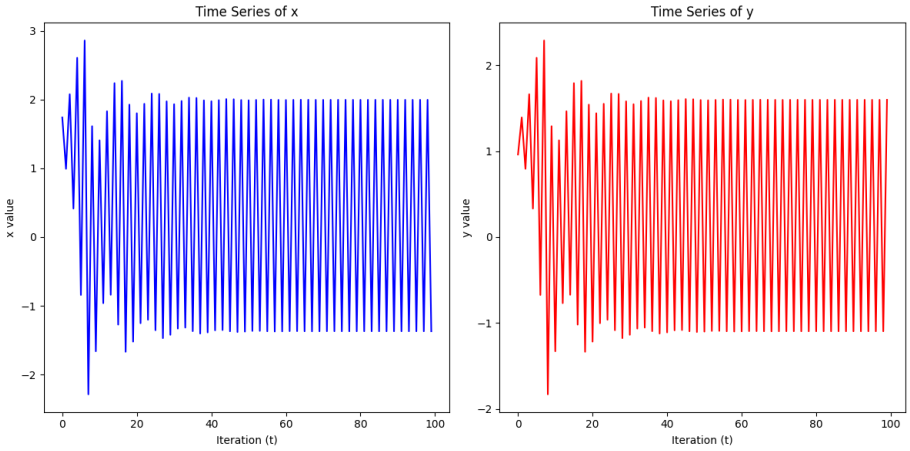




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

- $(a, b) = (0.32, 0.8)$, $x_0 = 1.2$ and $x_0 = 1.2$.

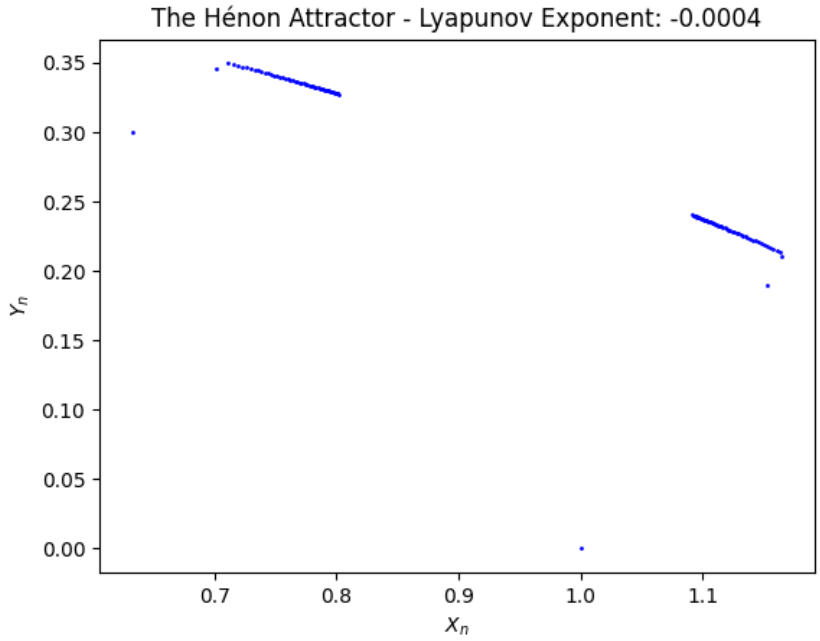


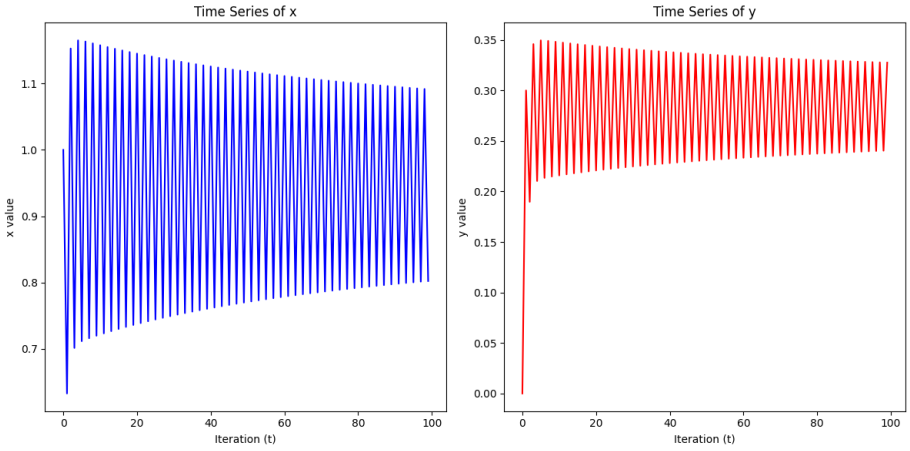


We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

We deduct that by raising the values of a,b the system transitions from periodic to chaotic. This can also be seen from the graphs and the Lyapunov exponent. However we observe that the initial conditions don't play a crucial role when the system is periodic. Small changes on them don't affect that much the shape of the plots and also don't affect the system's condition(chaotic or periodic) as seen in the first four plots of this exercise. But when the system is chaotic we can observe that Small changes on initial conditions do affect the shape of the plots. Also the affect the system's behaviour on chaos as observed by the plots.

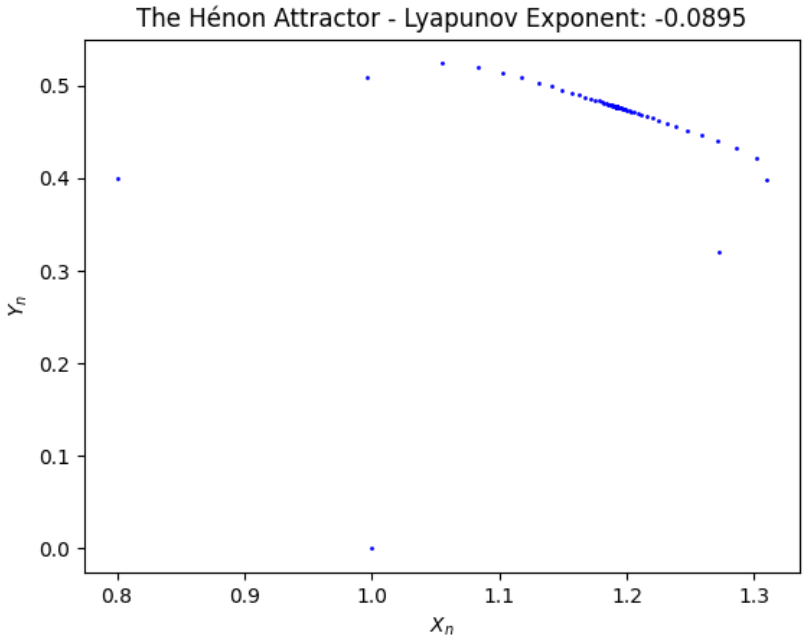
– $(a, b) = (0.3675, 0.3)$, $x_0 = 0$ and $x_0 = 0$.

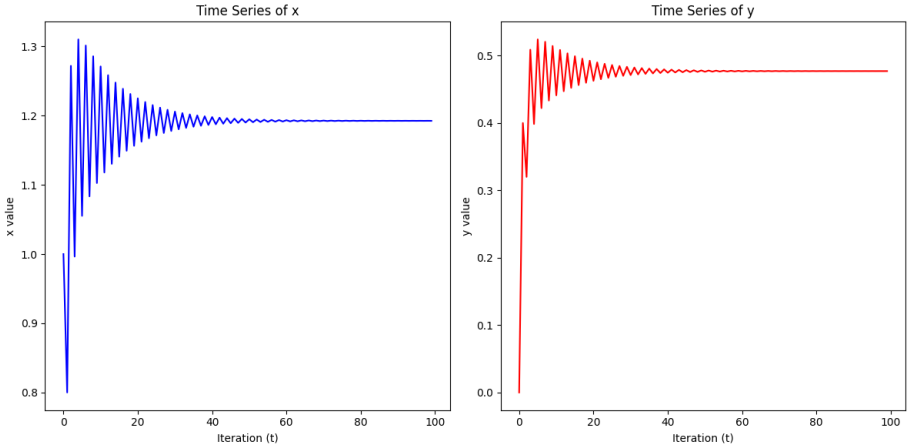




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots. It's on the verge of becoming chaotic which is confirmed by both plots(Attractor and time series).

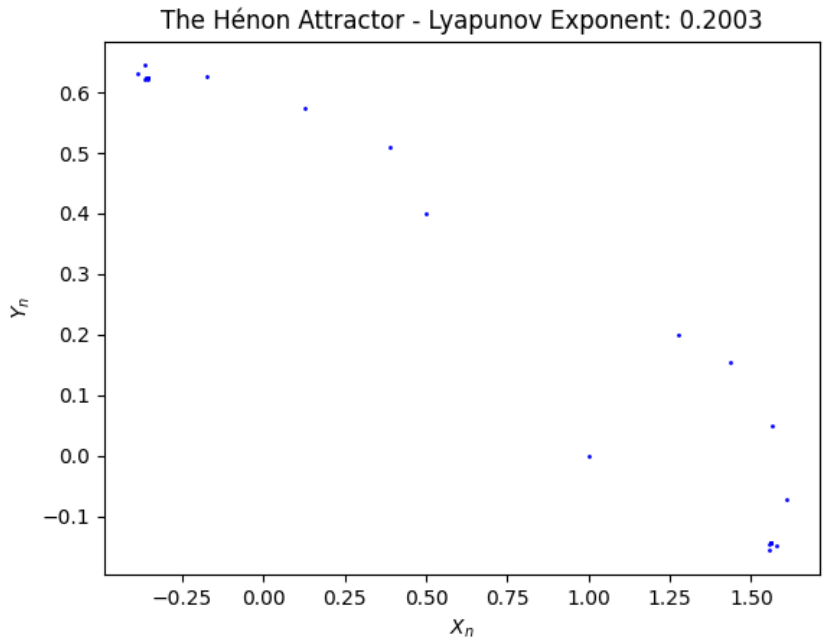
– $(a, b) = (0.2, 0.4)$, $x_0 = 0$ and $x_0 = 0$.

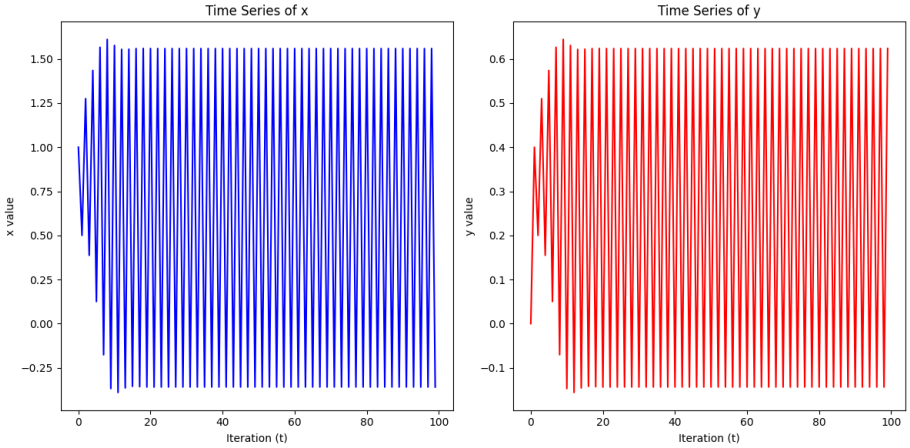




We observe some fixed points and periodic behaviour according to the the Lyapunov exponent and the time series plots.

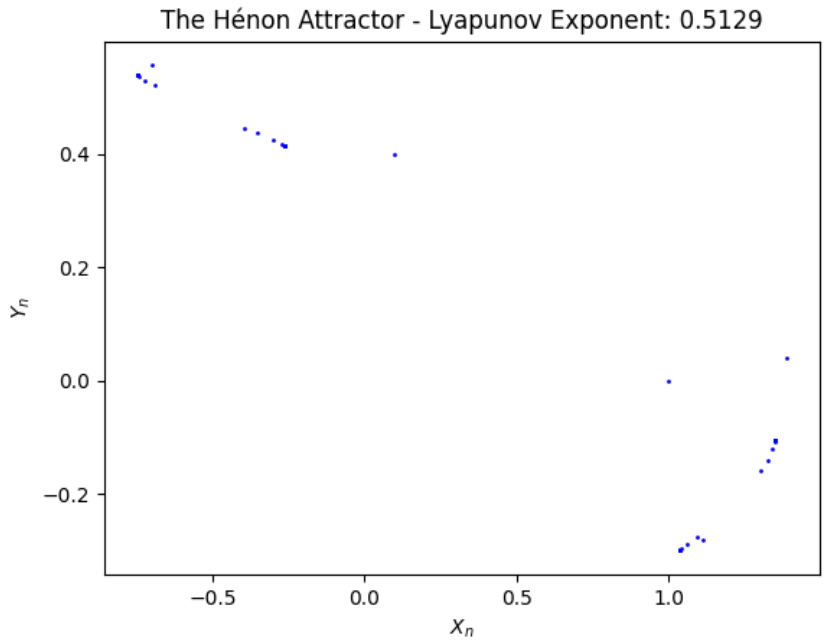
– $(a, b) = (0.5, 0.4)$, $x_0 = 0$ and $x_0 = 0$.

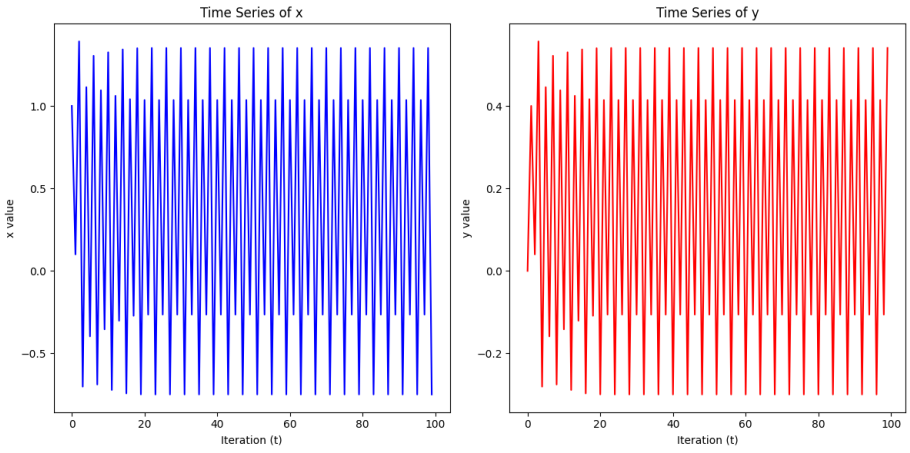




We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

– $(a, b) = (0.9, 0.4)$, $x_0 = 0$ and $x_0 = 0$.





We observe chaos which is confirmed by the the Lyapunov exponent and the time series plots.

We observe that by raising the values of a the system becomes more chaotic.

```
import numpy as np
from matplotlib import pyplot as plt

def henon_map(xn, yn, a, b):
    return yn + 1 - (a * xn**2), b*xn

def jacobian(xn, a, b):
    return np.array([[-2*a*xn, 1],
                     [b, 0]])

def lyapunov_exponent(jacobians):
    lyapunov_sum = 0
    for j in jacobians:
        eigenvalues = np.linalg.eigvals(j)
        max_eigenvalue = np.abs(eigenvalues).max()
        if max_eigenvalue == 0:
            max_eigenvalue += 1e-10
        lyapunov_sum += np.log(max_eigenvalue)
    return lyapunov_sum / len(jacobians)

def plot_map(x, y, lyap_exp):
    plt.scatter(x, y, c="blue", s=1)
    plt.xlabel(r"$X_n$")
    plt.ylabel(r"$Y_n$")
    plt.title(f"The H\xe9non Attractor - Lyapunov Exponent: {
                                    lyap_exp:.4f}")
    plt.show()

def plot_time_series(t_values, x_values, y_values):
    plt.figure(figsize=(12, 6))

    # Plot x vs t
    plt.subplot(1, 2, 1)
    plt.plot(t_values, x_values, c="blue")
    plt.title('Time Series of x')
    plt.xlabel('Iteration (t)')
    plt.ylabel('x value')

    # Plot y vs t
    plt.subplot(1, 2, 2)
    plt.plot(t_values, y_values, c="red")
    plt.title('Time Series of y')
    plt.xlabel('Iteration (t)')
```

```
    plt.ylabel('y value')

    plt.tight_layout()
    plt.show()

def main():
    # Set initial parameters and iterations
    a, b = 0.3, 0.4  # Example parameter values for chaotic
                                    behavior
    x, y = 0, 0  # Initial conditions
    iterations = 10000  # Number of iterations for the attractor
                                     plot
    time_series_iterations = 100  # Number of iterations for the
                                     time series plot

    xt, yt = [], []
    jacobians = []
    t_values = list(range(time_series_iterations))  # Store time
                                     steps for the time series

    # Perform the iterations
    for i in range(iterations):
        xn, yn = henon_map(x, y, a, b)
        jacobians.append(jacobian(xn, a, b))
        if i < time_series_iterations:
            # Store only the first 100 iterations for the time
                                            series
            xt.append(xn)
            yt.append(yn)
        x, y = xn, yn

    lyap_exp = lyapunov_exponent(jacobians)

    # Plot the Hénon map with Lyapunov exponent
    plot_map(xt, yt, lyap_exp)

    # Plot the time series using only the first 100 points
    plot_time_series(t_values, xt, yt)

# Call the main function
main()
```

4 PROBLEM-04

– Logistic sigmoid function (logsig). The logsig function is given by $S = \frac{1}{1+e^{-x}}$. The derivative S' is found as follows:

$$
\begin{aligned}
S' &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) \\
&= \frac{d}{dx}\left((1+e^{-x})^{-1}\right) \\
&= -(1+e^{-x})^{-2}\cdot(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2}
\end{aligned}
$$

Using S, we rewrite it as $S' = S \cdot (1 - S)$ since $S = \frac{1}{1+e^{-x}}$.

– Hyperbolic tangent function (tansig). The tansig function is given by $S = \tanh(x)$. Its derivative S' is calculated as follows:

$$
\begin{aligned}
S' &= \frac{d}{dx}\left(\tanh(x)\right) \\
&= \frac{d}{dx}\left(\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\right) \\
&= \frac{(e^{x}+e^{-x})(e^{x}+e^{-x}) - (e^{x}-e^{-x})(e^{x}-e^{-x})}{(e^{x}+e^{-x})^2} \\
&= 1 - \frac{(e^{x}-e^{-x})^2}{(e^{x}+e^{-x})^2} \\
&= 1 - S^2
\end{aligned}
$$

Therefore, $S' = 1 - S^2$, which is the derivative of the tansig function in terms of the function itself.

– Swish function. For the Swish function $S = \frac{x}{1+e^{-x}}$, its derivative S' is:

$$
\begin{aligned}
S' &= \frac{d}{dx}\left(\frac{x}{1+e^{-x}}\right) \\
&= \frac{1\cdot(1+e^{-x}) - x\cdot(-e^{-x})}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}+x\cdot e^{-x}}{(1+e^{-x})^2} \\
&= \frac{x\cdot e^{-x}}{(1+e^{-x})^2} + \frac{1}{1+e^{-x}} \\
&= \frac{x}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} + \frac{1}{1+e^{-x}}
\end{aligned}
$$

We can rewrite it using S as:

$$S' = S \cdot \sigma(-x) + \sigma(x)$$

where $\sigma(x) = \frac{e^{-x}}{1+e^{-x}}$ is the sigmoid function.

– Modified Swish-like function. For the Modified Swish-like function $S = x \cdot \tanh(\ln(1+e^x))$, its derivative S' is:

$$\begin{aligned} S' &= \frac{d}{dx}\left(x \cdot \tanh(\ln(1+e^x))\right) \\ &= \tanh(\ln(1+e^x)) + x \cdot \left(1 - \tanh^2(\ln(1+e^x))\right) \cdot \frac{e^x}{1+e^x} \end{aligned}$$

We can rewrite it using S as:

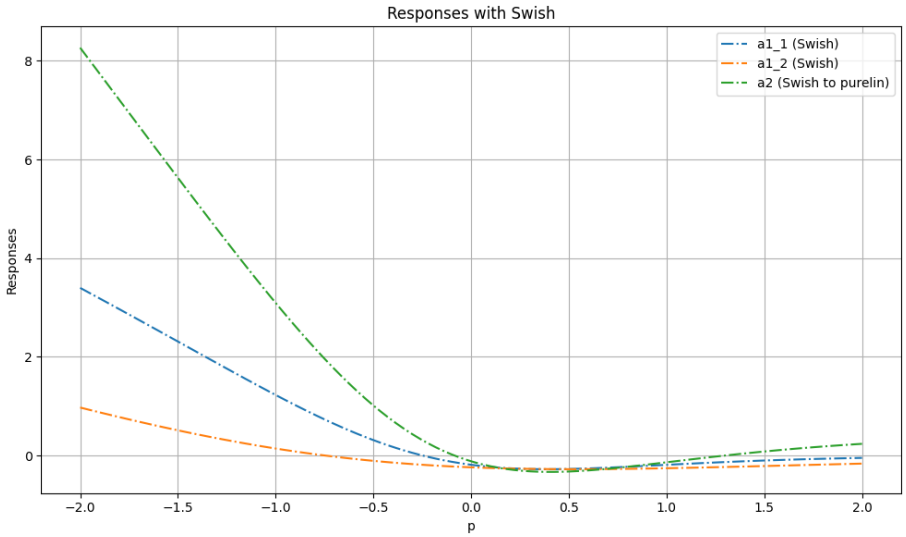
$$S' = \frac{S}{x} + x \cdot (1 - \frac{S^2}{x^2}) \cdot \sigma(x) = \frac{S}{x} + (1 - \frac{S^2}{x^2}) \cdot Swish$$

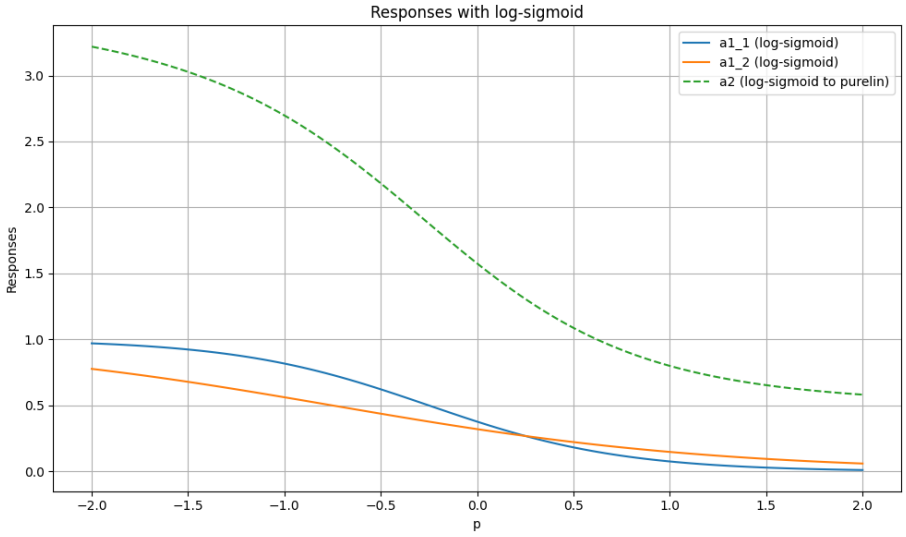
where $\sigma(x) = \frac{e^x}{1+e^x}$ is the sigmoid function.

5 PROBLEM-05

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

# Define the activation functions
def logsig(x):
    return 1 / (1 + np.exp(-x))

def purelin(x):
    return x

def swish(x):
    return x * logsig(x)

# Define the input range and network parameters
p = np.linspace(-2, 2, 100)
w1_1, w2_1, b1_1, b2_1 = -2.0, -1.0, -0.5, -0.75
w1_2, w2_2, b2 = 2.0, 1.0, 0.5

# Calculate the network outputs using the log-sigmoid activation
                                    function
n1_1 = w1_1 * p + b1_1
a1_1 = logsig(n1_1)
n1_2 = w2_1 * p + b2_1
a1_2 = logsig(n1_2)
```

```
n2 = w1_2 * a1_1 + w2_2 * a1_2 + b2
a2 = purelin(n2)

# Plot for log-sigmoid activation function
plt.figure(figsize=(10, 6))
plt.plot(p, a1_1, label='a1_1 (log-sigmoid)')
plt.plot(p, a1_2, label='a1_2 (log-sigmoid)')
plt.plot(p, a2, label='a2 (log-sigmoid to purelin)', linestyle='
                                --')
plt.title('Responses with log-sigmoid')
plt.xlabel('p')
plt.ylabel('Responses')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig('log_sigmoid_responses.png')  # Save the figure for
                                log-sigmoid
plt.show()

# Calculate the network outputs using the Swish activation
                                function
a1_1_swish = swish(n1_1)
a1_2_swish = swish(n1_2)
n2_swish = w1_2 * a1_1_swish + w2_2 * a1_2_swish + b2
a2_swish = purelin(n2_swish)

# Plot for Swish activation function
plt.figure(figsize=(10, 6))
plt.plot(p, a1_1_swish, label='a1_1 (Swish)', linestyle='-.')
plt.plot(p, a1_2_swish, label='a1_2 (Swish)', linestyle='-.')
plt.plot(p, a2_swish, label='a2 (Swish to purelin)', linestyle='
                                -.')
plt.title('Responses with Swish')
plt.xlabel('p')
plt.ylabel('Responses')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig('swish_responses.png')  # Save the figure for Swish
plt.show()
```





The plots above show the activations and outputs for both the log-sigmoid and swish activation functions across a range of input values from -2 to 2.

6 PROBLEM-06

This problem is addressing the impact of batch size on the performance of gradient descent optimization algorithms. We report the observations below:

If using a single data point per iteration (stochastic gradient descent with batch size 1) works better than using the entire dataset in each iteration (gradient descent with batch size equal to the dataset size), it means the model is learning effectively from random samples in each step (converges using random subsets at every iteration). Using a small but larger batch size(mini-batching), strikes a balance. It reduces the randomness in gradient calculations while still being less resource-intensive than using the full dataset. This is why a batch size of one-tenth of the dataset(nb = n/10) often gives better results than just one(nb = 1). However, as the batch size approaches the full dataset size, the process becomes as costly as it is using the entire dataset each time. It also risks repeating the same data patterns too often, which can worsen the method's performance. This is likely why increasing the batch size beyond a tenth of the dataset size leads to poorer results.

7 PROBLEM-07

Let's see how the activation function S(x) can approximate different known activation functions by tweaking its hyperparameters k, L, and m.

Hyperparameters:

- **k:** Controls the growth rate of the function for x > L.
- **L:** acts as a boundary value that dictates where the function changes its form and behavior.
- **m:** Affects the slope of the function within the $-L \leq x \leq L$ region.

- **Part A**

  - **Swish:** The Swish function can be approximated by S(x) for x > L when k = 1, as Swish behaves linearly for large positive x. For $-L < x < L$, we need S(x) to be roughly $\frac{x}{1+e^{-z}}$, which is the Swish function. To achieve this, m should be tuned such that the sigmoid component of Swish is approximated by the rational part of S(x). This would likely involve setting m such that $(L+X)^m/(L+X)^m + (L-X)^m$ is similar to the sigmoid function. We set m = 1, L = 3. So for x > 3 we have $S(x) = x$, and for x < -3 we have $S(x) = 0$. Now for $|x| \leq 3$:
    We Define the functions:
    $f(x) = \frac{x}{1+e^{-x}}$.
    $g(x) = \frac{x^2+3x}{6}$
    We Define the difference function:
    $h(x) = f(x) - g(x)$.

We Calculate the first derivative of the difference function:
$h'(x) = \frac{d}{dx}h(x) = \frac{d}{dx}\left(\frac{x}{1+e^{-x}}\right) - \frac{d}{dx}\left(\frac{x^2+3x}{6}\right)$.

Since h'(x) is complex, we would typically use numerical methods to find the roots. However, for a rough analysis, we can examine the behavior of h(x) by plotting or analyzing its components.

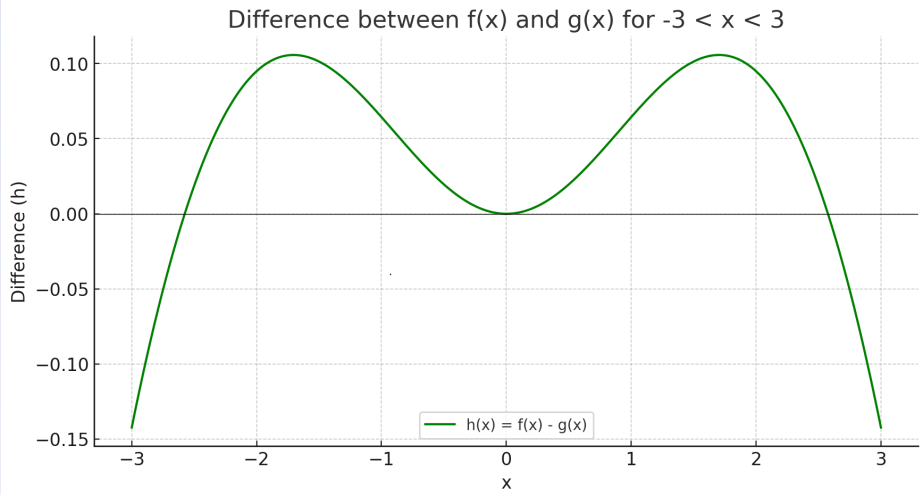
The first derivative of f(x) is:
$f'(x) = \frac{d}{dx}\left(\frac{x}{1+e^{-x}}\right) = \frac{(1+e^{-x})-xe^{-x}}{(1+e^{-x})^2}$.

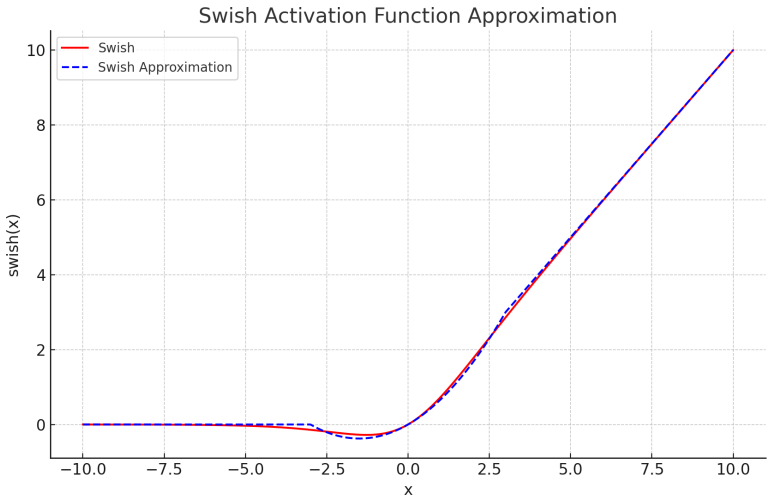
The first derivative of g(x) is:
$g'(x) = \frac{d}{dx}\left(\frac{x^2+3x}{6}\right) = \frac{2x+3}{6}$

The derivatives are then subtracted to find h'(x). We're looking for where this derivative is zero to find critical points, which indicate where the functions are closest or where their difference is changing direction.

We Evaluate the difference at the boundaries of the interval to ensure that they are close in magnitude.
h(-3) = f(-3) - g(-3) = −0.1423.
h(3) = f(3) - g(3) = −0.1423.

We plot the difference and we deduct that the maximum difference is very small, which means that it's a very good approximation:





- **Sigmoid:** For −L < x < L, we want S(x) to approximate the sigmoid curve, which requires L to be set to a value that captures the significant part of the sigmoid curve's transition from 0 to 1, while m should be adjusted to control the steepness of the transition. We set k = 0, L = 4.5, m = 2. So for X > 4.5 and x < -4.5 we have $S(x) = 0$. Now for $|x| \leq 4.5$:
  We Define the functions:
  $f(x) = \frac{1}{1+e^{-x}}$.
  $g(x) = \frac{(4.5+x)^2}{(4.5+x)^2+(4.5-x)^2}$
  We Define the difference function:
  $h(x) = f(x) - g(x)$.

  We Calculate the first derivative of the difference function:
  $h'(x) = \frac{d}{dx}h(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) - \frac{d}{dx}\ \frac{(4.5+x)^2}{(4.5+x)^2+(4.5-x)^2}\Big)$.

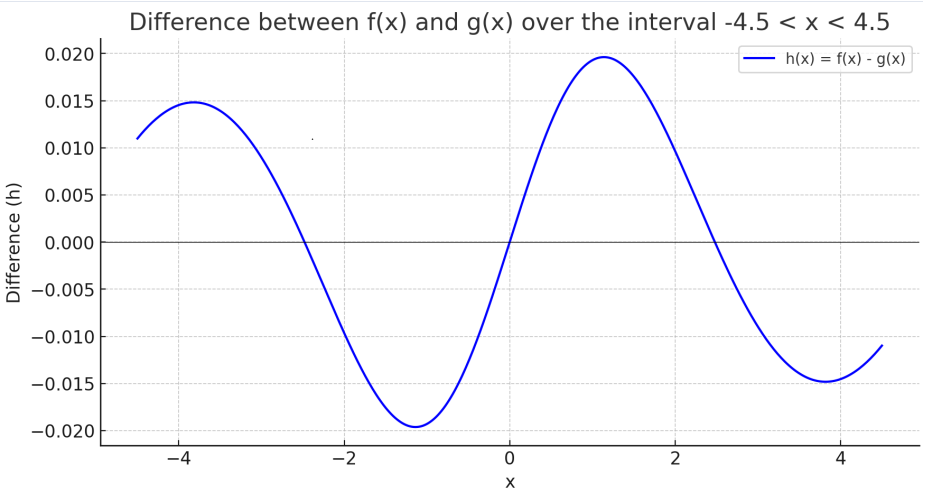
  Since h'(x) is complex, we would typically use numerical methods to find the roots. However, for a rough analysis, we can examine the behavior of h(x) by plotting or analyzing its components.

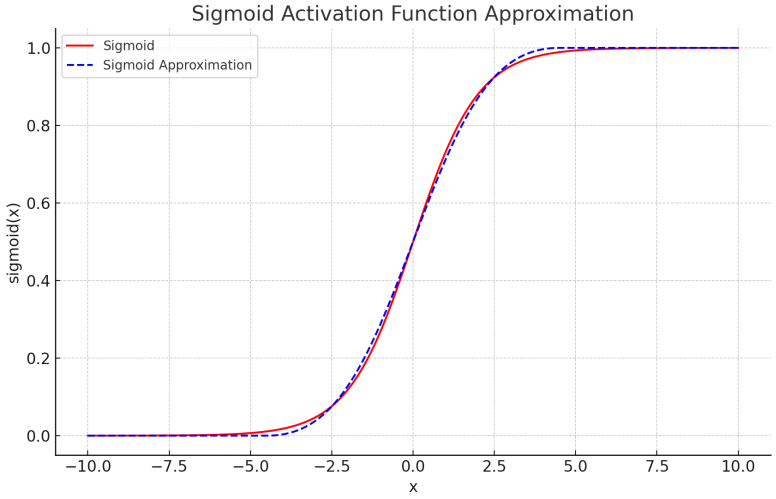
  The first derivative of f(x) is:
  $f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$..

  The first derivative of g(x) is:
  $g'(x) = -\frac{0.1975x(0.2222x+1)^2}{((1-0.2222x)^2+(0.2222x+1)^2)^2} + \frac{2.0x+9.0}{20.25((1-0.2222x)^2+(0.2222x+1)^2)}$.
  The derivatives are then subtracted to find h'(x). We're looking for where this derivative is zero to find critical points, which indicate where the functions are closest or where their difference is changing direction.

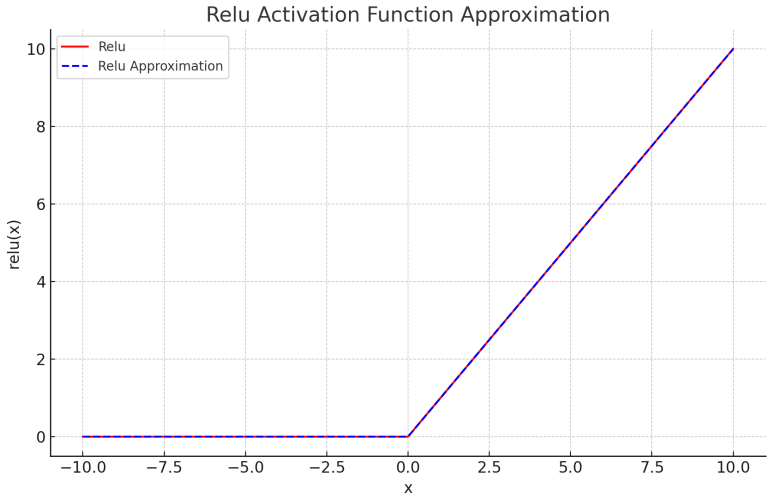
We Evaluate the difference at the boundaries of the interval to ensure that they are close in magnitude.
h(-4.5) = f(-4.5) - g(-4.5) = 0.0110. h(4.5) = f(4.5) - g(4.5) = −0.0110.

We plot the difference and we deduct that the maximum difference is very small, which means that it's a very good approximation:

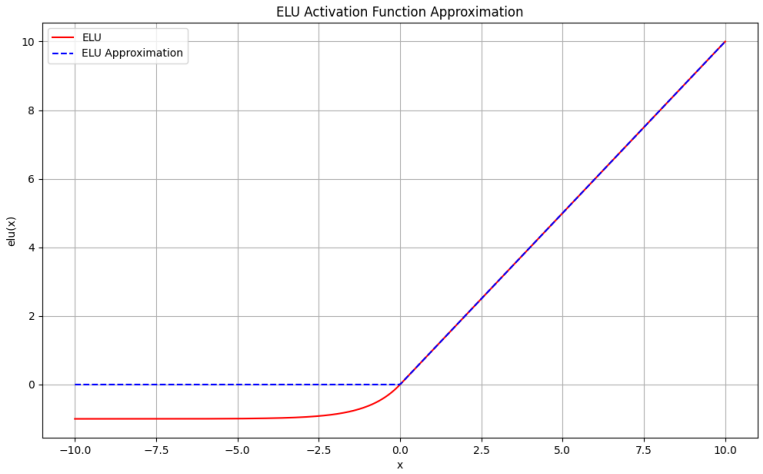




- **ReLU (Rectified Linear Unit):** ReLU is zero for negative values and linear with a slope of 1 for positive values. Set k=1, L = 0, and m doesn't matter as x will not be within $-L \leq x \leq L$ for the ReLU approximation. For $x \leq 0$ S(x) becomes $S(x) = x$ and for x > 0 it is $S(x) = 0$ which is exactly the ReLu activation function and also confirmed graphically.



- **ELU (Exponential Linear Unit):** ELU has a negative saturation for x<0 and linear with a slope of 1 for positive values. Set k = 1 to mimic the linear slope, L = 0, and m doesn't matter as x will not be within $-L \leq x \leq L$. It cannot be approximated efficiently because for values less than 0 it needs to mimic $e^x - 1$ and that is impossible because s(x) is bounded to be 0 for values less than -L. For $x \leq 0$ S(x) becomes $S(x) = x$ and for x > 0 it is $S(x) = 0$.



– **Part B**
We Calculate the derivatives of S(x) with respect to x, k, L,m:

$$\frac{\partial S}{\partial x} = \begin{cases} k \cdot x^{k-1} \\ \frac{\partial S}{\partial x} = \frac{(k \cdot x^{k-1} \cdot (L+x)^m + m \cdot x^k \cdot (L+x)^{m-1})((L+x)^m + (L-x)^m) - x^k \cdot (L+x)^m (m \cdot (L+x)^{m-1} - m \cdot (L-x)^{m-1})}{((L+x)^m + (L-x)^m)^2} \\ 0 \end{cases}$$

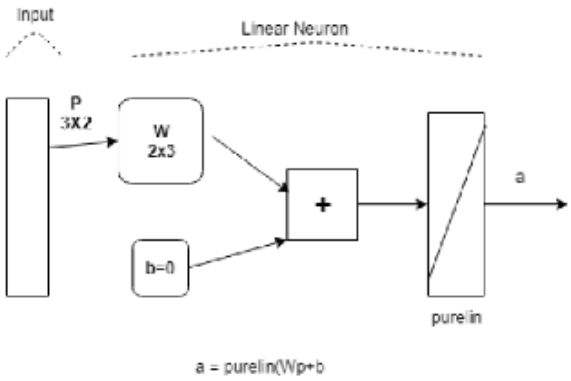
$$\frac{\partial S}{\partial k} = \begin{cases} x^k \cdot \ln(x) & \text{if } x > L \\ \frac{\partial S}{\partial k} = \frac{x^k \cdot \ln(x) \cdot (L+x)^m ((L+x)^m + (L-x)^m)}{((L+x)^m + (L-x)^m)^2} & \text{if } |x| \leq L \\ 0 & \text{if } x < -L \end{cases}$$

$$\frac{\partial S}{\partial L} = \begin{cases} 0 & \text{if } x > L \\ \frac{\partial S}{\partial L} = \frac{(m \cdot x^k \cdot (L+x)^{m-1})((L+x)^m + (L-x)^m) - x^k \cdot (L+x)^m (m \cdot (L+x)^{m-1} + m \cdot (L-x)^{m-1})}{((L+x)^m + (L-x)^m)^2} & \text{if } |x| \leq L \\ 0 & \text{if } x < -L \end{cases}$$

$$\frac{\partial S}{\partial m} = \begin{cases} 0 & \text{if } x > L \\ \frac{\partial S}{\partial m} = \frac{x^k \cdot (L+x)^m \cdot \ln(L+x)((L+x)^m + (L-x)^m) - x^k \cdot (L+x)^m ((L+x)^m \cdot \ln(L+x) + (L-x)^m \cdot \ln(L-x))}{((L+x)^m + (L-x)^m)^2} & \text{if } |x| \leq L \\ 0 & \text{if } x < -L \end{cases}$$

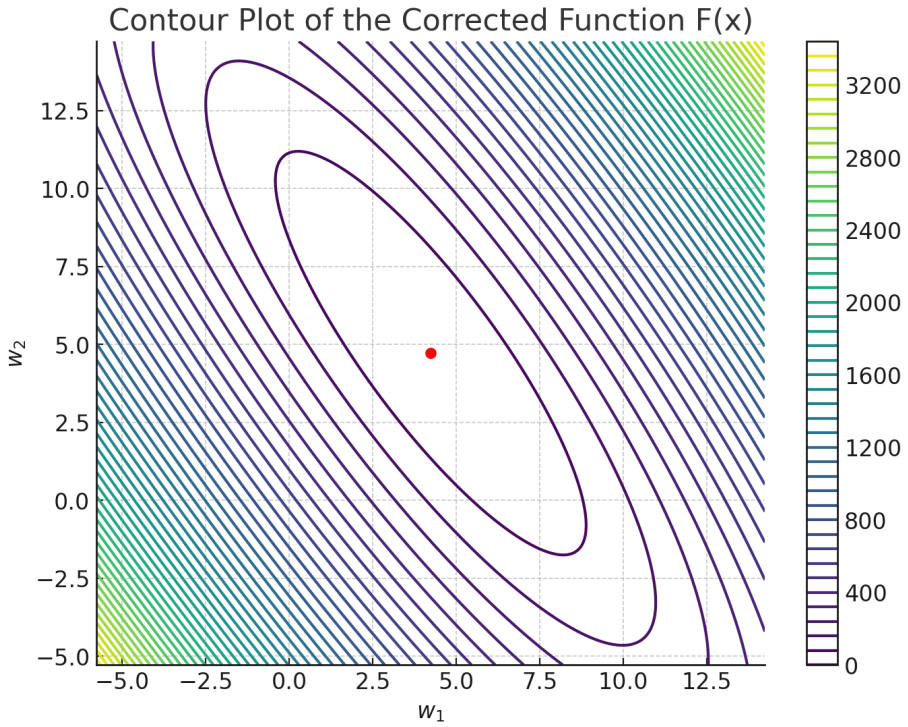
8 PROBLEM-08

– **Part A**



– **Part B**
Let's find first the MSE index: $F(x) = c - 2x^T h + x^T Rx$, x where $c = E[t^2], h = E[tz], R = E[zz^T]$.
$c = E[t^2] = 26^2 \times 0.2 + 26^2 \times 0.7 + (-26)^2 \times 0.1 = 676.$
$h = E[tz] = 0.2 \times 26 \times \begin{bmatrix} 2 \\ 4 \end{bmatrix} + 0.7 \times 26 \times \begin{bmatrix} 4 \\ 2 \end{bmatrix} + 0.1 \times (-26) \times \begin{bmatrix} -2 \\ -2 \end{bmatrix} = \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix}.$
$R = E[zz^T] = p_1 \times p_1^T \times (Prob1) + p_2 \times p_2^T \times (Prob2) + p_3 \times p_3^T \times (Prob3) = \begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix}.$
So, the MSE index is equal to $F(x) = c - 2x^\top h + x^\top Rx =$
$676 - 2 \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix} + \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} =$
$12.4 \times w_1^2 + 15.2 \times w_1 \times w_2 - 176.8 \times w_1 + 6.4 \times w_2^2 - 124.8 \times w_2 + 676.$

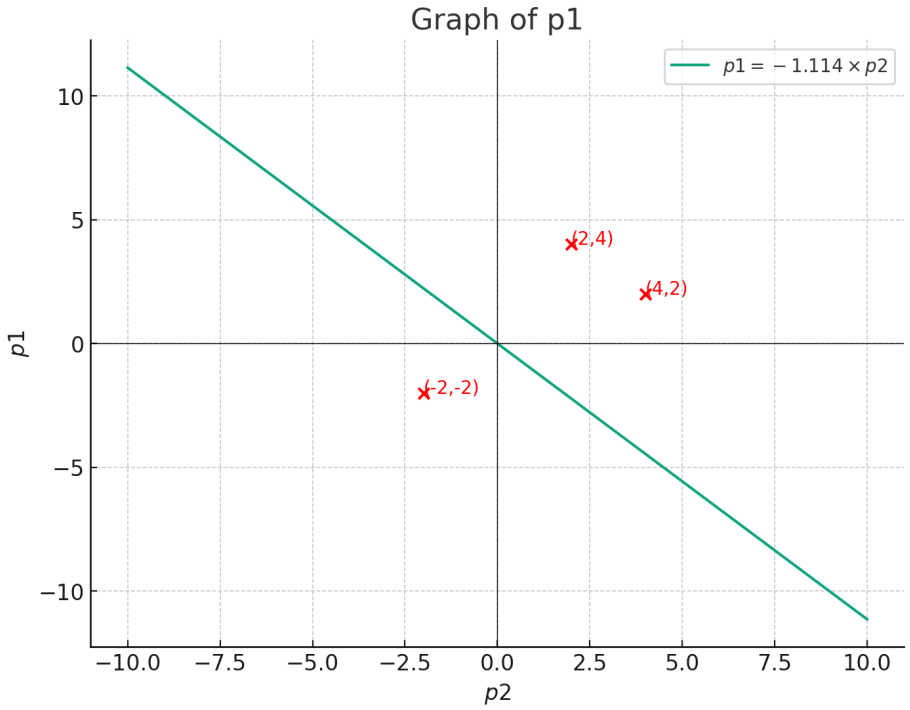
To find the center of the contour, we need to solve for the minimum w*.
$w* = R^{-1} \times h = \begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix}^{-1} \times \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix} = \begin{bmatrix} \frac{8}{27} & -\frac{19}{54} \\ -\frac{19}{54} & \frac{31}{54} \end{bmatrix} \times \begin{bmatrix} 88.4 \\ 62.4 \end{bmatrix} = \begin{bmatrix} 4.237 \\ 4.719 \end{bmatrix}$. We can verify it below through its contour plot:



– **Part C**

For bias = 0 and $w_{minMSE} = \begin{bmatrix} 4.237 \\ 4.719 \end{bmatrix}$ we solve $w^T \times p + b = 0 => w^T \times p = 0 =>$ $4.237 \times p1 = -4.719 \times p2 => p1 = -1.114 \times p2$.

Let's visualize the optimal decision boundary (for min MSE):



- **Part D**
  First, we need to find the eigenvalues of R.
  R = $\begin{bmatrix} 12.4 & 7.6 \\ 7.6 & 6.4 \end{bmatrix}$ so $\lambda 1 \approx 1.23$ and $\lambda 2 \approx 17.57$. So, from the slides
  $0 < a < 1.23/17.57 = 0.07$
  where a is the maximum stable learning rate. Now, if target values to change to 11 and -11, there will be no change to $R = E[zz^T]$. Sincerely, there will be no changes to the learning rate.

- **Part E**
  Given the weight vector W and input vector p, the initial output is calculated as:

$$W \times p = 0 \cdot p = 0 = \text{result}$$

  The error is then calculated as:

$$\text{error} = \text{target} - \text{result} = 26 - 0 = 26$$

From the slides, the LMS algorithm update rule for weights is given by:

$$\begin{aligned} W(k+1) &= W(k) + 2\alpha \cdot e(k)p^T \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 2 \cdot 0.05 \cdot 26 \cdot \begin{bmatrix} 2 \\ 4 \end{bmatrix} \\ &= \begin{bmatrix} 5.2 \\ 10.4 \end{bmatrix} \end{aligned}$$

The LMS algorithm update rule for bias is given by:

$$\begin{aligned} b(k+1) &= b(k) + 2 \cdot \alpha \cdot e(k) \\ &= 0 + 2 \cdot 0.05 \cdot 26 \\ &= 2.6 \end{aligned}$$

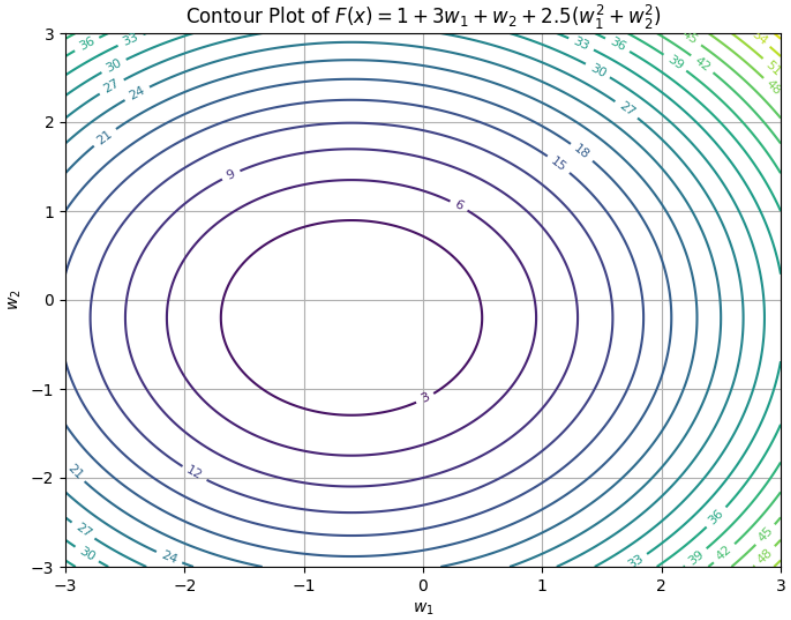
9 PROBLEM-09

- **Part A**
  Since the vectors are equiprobable we have Prob1 = Prob2 = 0.5.
  $F(x) = c - 2x^\top h + x^\top Rx$

  where
  $c = E[t^2]$,
  $h = E[tz]$,
  $R = E[zz^\top]$

  and
  $c = E[t^2]$,
  $h = E[tz] = 0.5 \cdot (-1) \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 0.5 \cdot 1 \cdot \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix}$,
  $R = E[zz^\top] = p_1 p_1^\top \cdot \text{Prob}_1 + p_2 p_2^\top \cdot \text{Prob}_2 =$
  $= \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \cdot 0.5 + \begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix} \cdot 0.5 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}$

  Therefore
  $F(x) = c - 2x^\top h + x^\top Rx =$
  $1 - 2 \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix} + \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} =$
  $1 + 3w_1 + w_2 + 2.5(w_1^2 + w_2^2)$

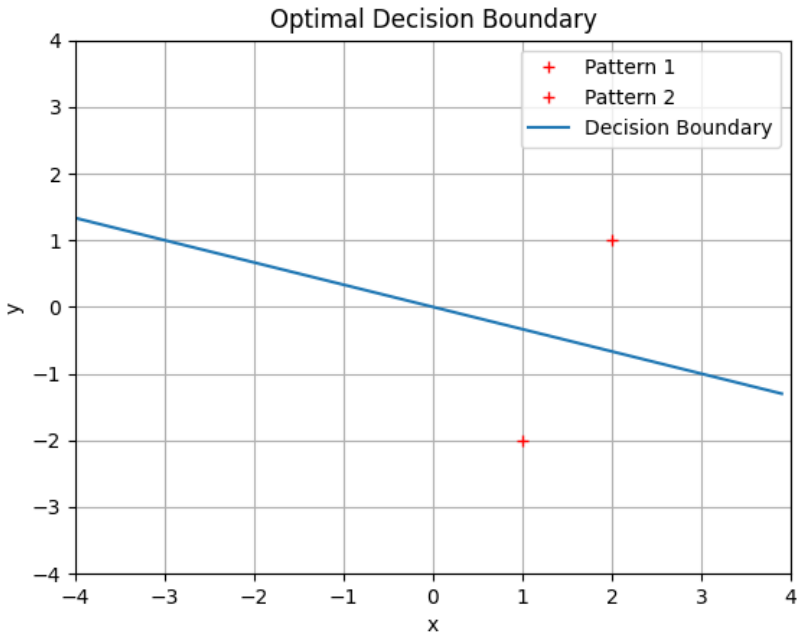



```
import numpy as np
import matplotlib.pyplot as plt

# Define the range for w1 and w2
w1 = np.linspace(-3, 3, 400)
w2 = np.linspace(-3, 3, 400)

# Create a grid of values
W1, W2 = np.meshgrid(w1, w2)

# Define the function F(x)
F = 1 + 3*W1 + W2 + 2.5*(W1**2 + W2**2)

# Create the contour plot
plt.figure(figsize=(8, 6))
contour = plt.contour(W1, W2, F, levels=20, cmap='viridis')
plt.clabel(contour, inline=True, fontsize=8)
plt.title('Contour Plot of F(x) = 1 + 3w_{1} + w_{2} + 2.5(w_
                                {1}^2 + w_{2}^2)')
plt.xlabel('$w_{1}$')
plt.ylabel('$w_{2}$')
plt.grid(True)
plt.show()
```

– **Part B**
We need to find the optimal weights:

$$w^* = R^{-1} \cdot h = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -0.6 \\ -0.2 \end{bmatrix} \quad (11.4)$$

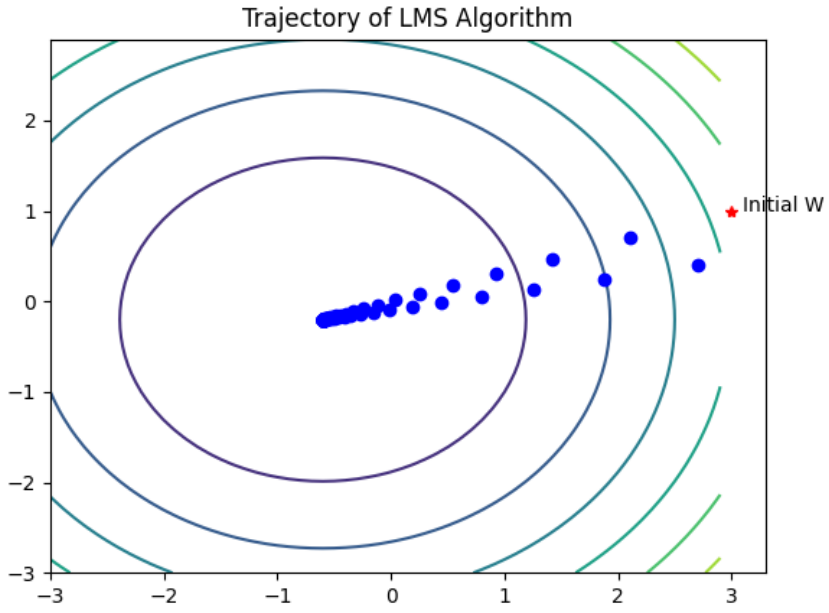
So, the optimal weights are $w_1 = -0.6$ and $w_2 = -0.2$.
It is clear, from the image that the optimal decision boundary separates the patterns into the appropriate categories.




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

# Input data and set parameters
P = np.array([[1, 2],[-2,1]])  # Reference patterns
W = np.array([-0.6, -0.2])        # Optimal weights

# Plot the reference patterns
plt.figure()
plt.plot(P[0, 0], P[1, 0], 'r+', label='Pattern 1')
plt.plot(P[0, 1], P[1, 1], 'r+', label='Pattern 2')
```

```
# Decision Boundary
x = np.arange(-4, 4, 0.1)
y = (-W[1] * x) / W[0]
plt.plot(x, y, label='Decision Boundary')

# Setting the axis limits and title
plt.axis([-4, 4, -4, 4])
plt.title('Optimal Decision Boundary')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

– **Part C**

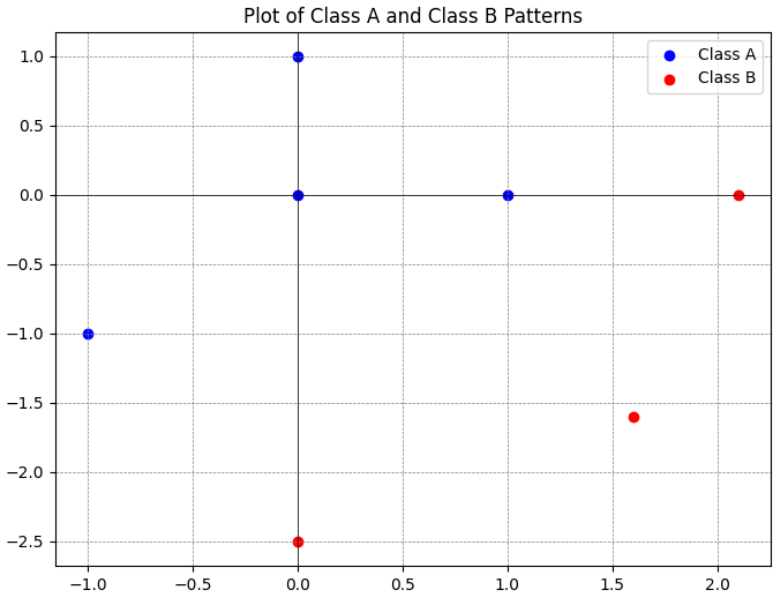



```
import numpy as np
import matplotlib.pyplot as plt

# Generating the meshgrid
W1, W2 = np.meshgrid(np.arange(-3, 3, 0.1), np.arange(-3, 3,
                                    0.1))
```

```
# Defining the function F
F = 1 + 3 * W1 + W2 + 2.5 * (W1**2 + W2**2)

# Plotting the contour
plt.contour(W1, W2, F)
plt.title('Trajectory of LMS Algorithm')

# Input data and set parameters
P = np.array([[1, -2], [2, 1]])  # reference patterns
T = [-1, 1]  # targets
alpha = 0.025  # learning rate

# Initialize weight
W = np.array([3, 1])

# Plot initial weight
plt.plot(W[0], W[1], 'r*')  # Initial position of weight
plt.text(W[0] + 0.1, W[1], f'Initial W')

# Training the ADALINE network without a bias
for step in range(71):
    for i in range(2):
        a = np.dot(W, P[:, i])
        e = T[i] - a
        W = W + 2 * alpha * e * P[:, i]

        print(f'Step {step}, Pattern {i}, Output: {W}')
        plt.plot(W[0], W[1], 'bo')  # Marking the new
                                            position of weights

plt.show()
```

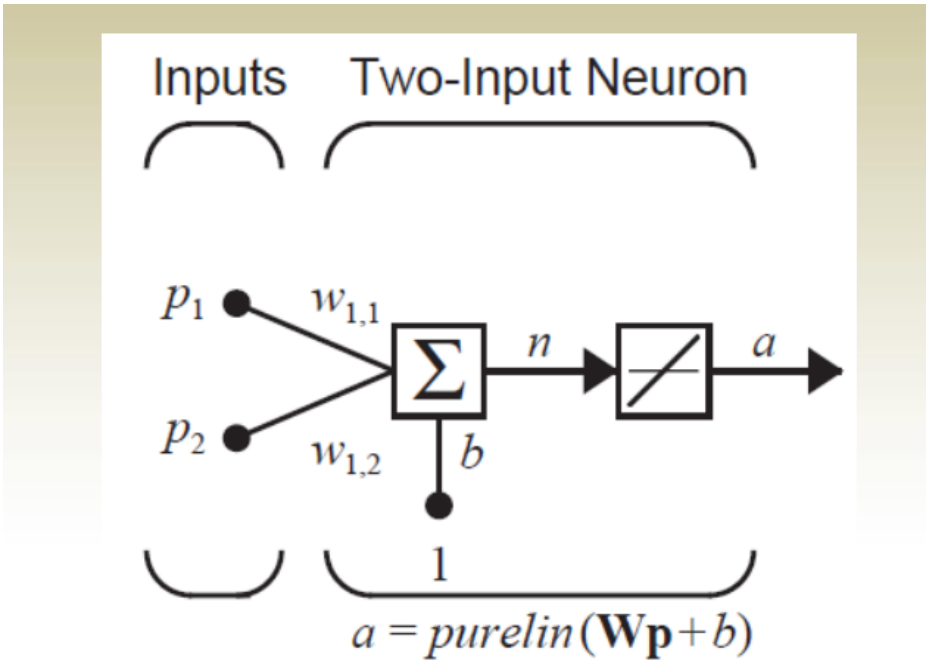
10 PROBLEM-10

– **Part A**
We can observe from the image that the categories are linearly seperable.




```python
import matplotlib.pyplot as plt

# Defining the points for each class
class_A = [(0, 0), (0, 1), (1, 0), (-1, -1)]
class_B = [(2.1, 0), (0, -2.5), (1.6, -1.6)]

# Unpacking the points into x and y coordinates for plotting
x_A, y_A = zip(*class_A)
x_B, y_B = zip(*class_B)

# Creating the plot
plt.figure(figsize=(8, 6))
plt.scatter(x_A, y_A, color='blue', label='Class A')
plt.scatter(x_B, y_B, color='red', label='Class B')

# Adding details to the plot
plt.title('Plot of Class A and Class B Patterns')
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.grid(color = 'gray', linestyle = '--', linewidth = 0.5)
plt.legend()
plt.show()
```

– **Part B**



We will use a single ADALINE neuron. P1 and P2 are the 2 seperate classes of patterns.

– **Part C**

We used the script below to produce the optimal weights and biases.

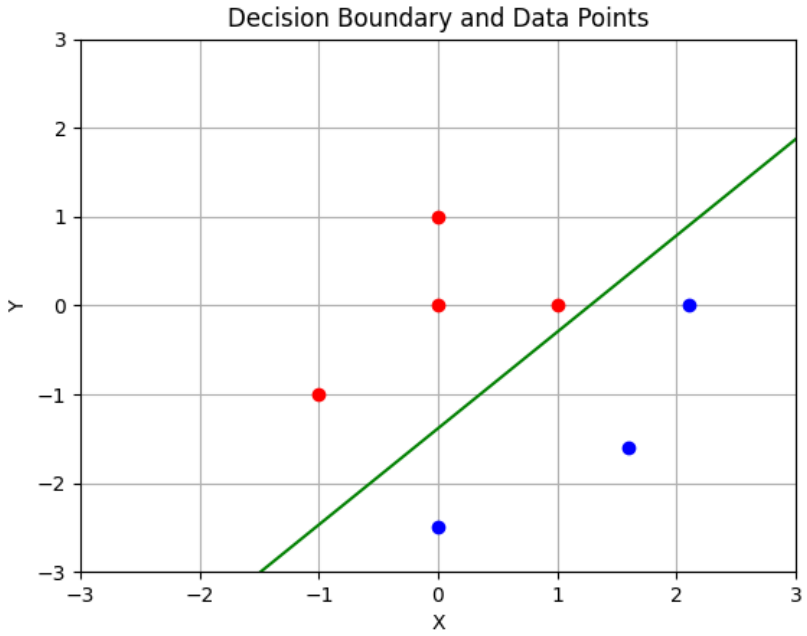
```
import numpy as np

def train_adaline(P, T, alpha, epochs):
    # Initialize weights and bias to 0.5
    W = np.array([0.5, 0.5])
    b = 0.5

    for _ in range(epochs):
        for i in range(len(T)):
            a = np.dot(W, P[i]) + b
            e = T[i] - a
            W = W + 2 * alpha * e * P[i]
            b = b + 2 * alpha * e  # Update bias
```

```
    return W, b


# Data points and labels
P = np.array([[0, 0], [0, 1], [1, 0], [-1, -1], [2.1, 0], [0,
                                    -2.5],
              [1.6, -1.6]])
T = np.array([-1, -1, -1, -1, 1, 1, 1])

# ADALINE training parameters
alpha = 0.01  # Learning rate
epochs = 1000  # Number of epochs

# Train the ADALINE
optimal_weights, optimal_bias = train_adaline(P, T, alpha,
                                    epochs)

print("Optimal weights:", optimal_weights)
print("Optimal bias:", optimal_bias)
```

which gave us the following output Optimal weights: [0.65739194 -0.60556104] Optimal bias: -0.8380181799345243. Finally, we used this script to calculate the optimal decision boundary so that we can confirm that the values are correct.

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

# Data points and labels
P = np.array([[0, 0], [0, 1], [1, 0], [-1, -1], [2.1, 0], [0,
                                    -2.5], [1.6, -1.6]])
T = np.array([-1, -1, -1, -1, 1, 1, 1])

# Optimal weights and bias
optimal_weights = np.array([0.65739194, -0.60556104])
optimal_bias = -0.8380181799345243

# Plot data points
for point, label in zip(P, T):
    if label == -1:
        plt.plot(point[0], point[1], 'ro')  # Class -1 as red
    else:
        plt.plot(point[0], point[1], 'bo')  # Class 1 as blue

# Decision boundary
x = np.linspace(-3, 3, 100)  # x range
y = (-optimal_weights[0] / optimal_weights[1]) * x - (
                                    optimal_bias /
                                    optimal_weights[1])  # y as
                                    a function of x
```

```
plt.plot(x, y, 'g-')  # Decision boundary in green

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Decision Boundary and Data Points')
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.grid()
plt.show()
```



11 PROBLEM-11

– **Part A**
True.
Given a fuzzy set S with membership function $\mu_S(x)$ for an element x, the set "Very S" is constructed by squaring the membership function:

$$\mu_{\text{Very S}}(x) = (\mu_S(x))^2.$$

Since $0 \leq \mu_S(x) \leq 1$ for all x in S, it follows that $(\mu_S(x))^2 \leq \mu_S(x)$. Therefore, $\mu_{\text{Very S}}(x) \leq \mu_S(x)$, which makes "Very S" a fuzzy subset of S.

– **Part B**
True.
Using the modifier ”more or less” results in the following transformation of the membership function:

$$\mu_{\text{More or Less S}}(x) = \sqrt{\mu_S(x)}.$$

Since the square root function is monotonically increasing for $0 \leq x \leq 1$, and $\sqrt{\mu_S(x)} \geq \mu_S(x)$, the original set S is a fuzzy subset of ”more or less S”.

– **Part C**
Impossible to say without additional information.
The sets ”not very S” and ”more or less S” are defined as:

$$\mu_{\text{Not Very S}}(x) = 1 - (\mu_S(x))^2,$$
$$\mu_{\text{More or Less S}}(x) = \sqrt{\mu_S(x)}.$$

Without specific values of $\mu_S(x)$, we cannot determine the subset relationship between ”not very S” and ”more or less S”.

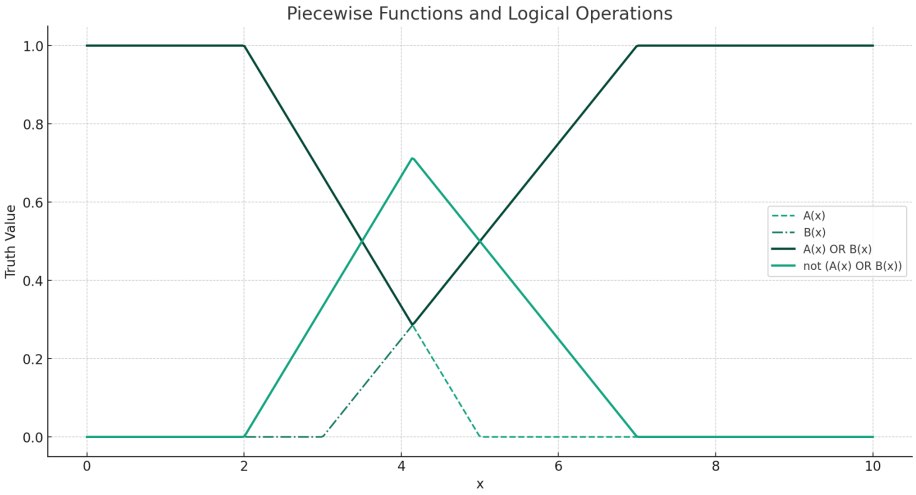
– **Part D**
Impossible to say without additional information.
The sets ”not more or less S” and ”very S” are defined as:

$$\mu_{\text{Not More or Less S}}(x) = 1 - \sqrt{\mu_S(x)},$$
$$\mu_{\text{Very S}}(x) = (\mu_S(x))^2.$$

Without further information on the membership function $\mu_S(x)$, we cannot analytically determine a subset relationship between ”not more or less S” and ”very S”.

12 PROBLEM-12

Lets visualize A(x), B(x), A(x) or B(x), not(A(x) or B(x)):



1. For $x \leq 2$:
   - $A(x)$ is 1, and $B(x)$ is 0.
   - Since $A(x)$ is true, the OR operation yields true.
   - Therefore, "not $(A(x)$ OR $B(x))$" is 0 because NOT inverts the truth value.
2. For $2 < x \leq 3$:
   - $A(x)$ starts decreasing from 1, and $B(x)$ remains 0.
   - The OR operation is solely determined by $A(x)$ since $B(x)$ is false.
   - "not $(A(x)$ OR $B(x))$" starts to increase from 0 as $A(x)$ decreases from 1.
3. For $3 < x < 5$:
   - $A(x)$ continues to decrease, and $B(x)$ starts to increase.
   - At $x = \frac{29}{7}$, $A(x)$ and $B(x)$ intersect, meaning they have the same truth value.
   - Before the intersection, "not $(A(x)$ OR $B(x))$" increases as $A(x)$ decreases, since $B(x)$ is lower than $A(x)$.
   - At the intersection, "not $(A(x)$ OR $B(x))$" reaches its maximum value that is above 0 but less than 1, because both $A(x)$ and $B(x)$ are true but neither is at full truth value.
   - After the intersection and up to $x < 5$, "not $(A(x)$ OR $B(x))$" starts to decrease as $B(x)$ becomes the dominant value in the OR operation.
4. For $5 \leq x < 7$:
   - $A(x)$ is 0, and $B(x)$ continues to increase towards 1.
   - The OR operation's truth value increases with $B(x)$, but since "not $(A(x)$ OR $B(x))$" inverts this, its value starts high after the intersection point and decreases towards 0 as $B(x)$ approaches its full truth value of 1.
   - "not $(A(x)$ OR $B(x))$" is therefore a decreasing function in this interval, starting from the maximum value just after the intersection and approaching 0 as x approaches 7.
5. For $x \geq 7$:

- $A(x)$ is 0, and $B(x)$ is 1.
- The OR operation yields true because of $B(x)$.
- Thus, "not $(A(x)$ OR $B(x))$" is 0 since NOT inverts the truth value of the OR operation.

In summary, "not $(A(x)$ OR $B(x))$" is 0 for $x \leq 2$ and $x \geq 7$. It increases from 0 just after $x = 2$, reaches a maximum value between 3 and 5 at $x = \frac{29}{7}$, and then decreases back towards 0 as x approaches 7. This maximum value is the highest truth value of "not $(A(x)$ OR $B(x))$" in the interval between 2 and 7.