Лабораторні роботи 3-5

Створено системою Doxygen 1.9.1

1 Звіт з лабораторних робіт №3-5

за дисципліною "Элементи хаотичної динаміки"
студента групи ПА-17-2
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2020-2021 навч.р.
Варіант 17

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/EoCD_Labs/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/08/EoCD_Labs

1.1 Постановка задачі

1.1.1 Лабораторна робота 3

Для такої системи диференціальних рівнянь знайти всі ізольовані особливі точки, встановити їх тип і побудувати фазовий портрет при декількох початкових даних.

$$\{ x_1'(t) = x_1(t)^2 - 2x_1(t)x_2(t) + x2(t)^2 - 9x_2'(t) = 4x_1(t)^2 + x_1x_2(t) + 4x_2(t)^2 - 18$$

1.1.2 Лабораторна робота 4

Побудувати біфуркаційну діаграму для наступних дискретних відображень і вказати значення параметра α, при якому відбувається перехід до хаосу. (Попередньо встановити відрізок I числової осі, який дане відображення $f(x, \alpha)$ відображає в себе: $f(I, \alpha) \in I$.)

$$x_{n+1} = \alpha x_n^2(1 - x_n^2),\ n = 0, 1, 2, \ldots$$

1.1.3 Лабораторна робота 5

Побудувати хаотичні атрактори для наступних 3-мірних систем автономних дифференціальних рівнянь.

$$\{ x'(t) = 3x(t) - y(t)^2 + z^2(t)y'(t) = -y(t) - 700z(t) + 10x(t)y(t)z'(t) = x(t) + 700y(t) - z(t) - 15x(t)z(t)$$

2 Алфавітний покажчик простору імен

2.1 Простір імен

Повний список просторів імен.

3 Покажчик файлв

3.1 Файли

Повний список файлів.

4 Опис простору імен

4.1 Простір імен lab3

Лабораторна робота 3.

Функції

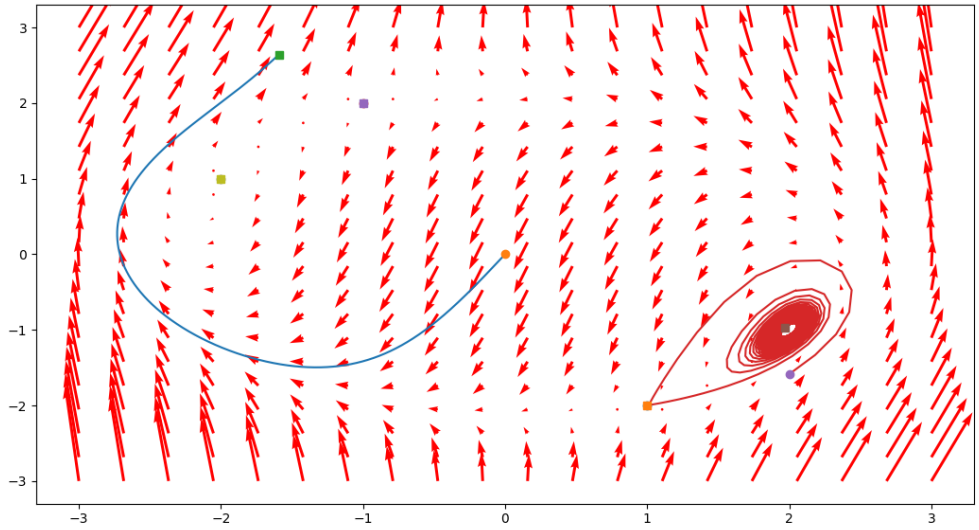
- np.ndarray calc (X, int t=0)
  Обчислює похідну системи по її значенням.
- None print_vector_field (np.ndarray x, np.ndarray y)
  Обчислює векторне поле у діапазонах [-3, 3] по обом осям.
- None print_trajectories (list starts)
  Будуємо траєкторії.

Змінні

- x = np.linspace(-3.0, 3.0, 20)
  Задаємо діапазон x для побудови векторного поля та кількість векторів.
- y = np.linspace(-3.0, 3.0, 20)
  Задаємо діапазон y для побудови векторного поля та кількість векторів.
- list starts
  Задаємо декілька траєкторій.

4.1.1 Детальний опис

Лабораторна робота 3.



4.1.2 Опис функцій

4.1.2.1 calc() np.ndarray lab3.calc (X, int t = 0)

Обчислює похідну системи по її значенням.

$$\{ x_1'(t) = x_1(t)^2 - 2x_1(t)x_2(t) + x2(t)^2 - 9x_2'(t) = 4x_1(t)^2 + x_1x_2(t) + 4x_2(t)^2 - 18$$

Аргументи

X	точка системи
t	час

Повертає

вектор похідної

Див. визначення в файлі lab3.py, рядок 39

```
00039 def calc(X, t: int = 0) -> np.ndarray:
00040     x1, x2 = X[0], X[1]
00041     return np.array([
```

```
00042         x1**2-2*x1*x2+x2**2-9,
00043         4*x1**2+x1*x2+4*x2**2-18,
00044     ])
00045
```

4.1.2.2 print_trajectories() None lab3.print_trajectories (list starts)

Будуємо траєкторії.

Аргументи

starts	змінна starts, яка задає усі траєкторії

Див. визначення в файлі lab3.py, рядок 62

```
00062 def print_trajectories(starts: list) -> None:
00063     # Обчислюємо траєкторії задані у змінній ‘starts‘
00064     for s, t in starts:
00065         # Задаємо діапазон часу [0, t] та ділить відрізок на 2000 частин.
00066         tspan = np.linspace(0, t, 2000)
00067         # Інтегрує систему до заданого часу ‘t‘ та початкових значень ‘s‘
00068         tr = odeint(calc, s, tspan)
00069         # Друкуєму саму траєкторію
00070         plt.plot(tr[:,0], tr[:,1])
00071         # Друкуємо початок траєкторії у вигляді кружечка
00072         plt.plot([tr[0,0]], [tr[0,1]], 'o')
00073         # Друкуємо кінець траєкторії у вигляді квадратика
00074         plt.plot([tr[-1,0]], [tr[-1,1]], 's')
00075
00076
00077 print_vector_field(x, y)
00078 print_trajectories(starts)
00079 # Друкуємо графік
00080 plt.show()
```

4.1.2.3 print_vector_field() None lab3.print_vector_field (np.ndarray x, np.ndarray y)

Обчислює векторне поле у діапазонах [-3, 3] по обом осям.

Аргументи

x	массив усіх можливих значень x
y	массив усіх можливих значень y

Див. визначення в файлі lab3.py, рядок 49

```
00049 def print_vector_field(x: np.ndarray, y: np.ndarray) -> None:
00050     X, Y = np.meshgrid(x, y)
00051     u, v = np.zeros(X.shape), np.zeros(Y.shape)
00052     NI, NJ = X.shape
00053     # Обчислює векторне поле по заданим значенням.
00054     for i in range(NI):
00055         for j in range(NJ):
00056             u[i, j], v[i, j] = calc([X[i, j], Y[i, j]])
00057     # Друкує на графік векторне поле.
00058     plt.quiver(X, Y, u, v, color='r')
00059
```

4.1.3 Опис змінних

4.1.3.1 starts list lab3.starts

Початкові значення
```
00001 = [
00002     ([0, 0], 0.6),
00003     ([2, -1.585], 100),
00004     ([-2.0, 1.0], 100),
00005     ([1, -2], 100),
00006     ([-1, 2], 100),
00007 ]
```

Задаємо декілька траєкторій.

Для кожної траєкторії задаємо початкові значення та час для траєкторії.

Загалом у нас 5 траєкторій:

- З початком у кординаті (0, 0) та часом 0.6
- З початком у кординаті (2, -1.585) та часом 100
- З початком у кординаті (-2.0, 1.0) та часом 100
- З початком у кординаті (1, -2) та часом 100
- З початком у кординаті (-1, 2) та часом 100

Див. визначення в файлі lab3.py, рядок 24

4.1.3.2 x lab3.x = np.linspace(-3.0, 3.0, 20)

Задаємо діапазон x для побудови векторного поля та кількість векторів.

Див. визначення в файлі lab3.py, рядок 12

4.1.3.3 y lab3.y = np.linspace(-3.0, 3.0, 20)

Задаємо діапазон y для побудови векторного поля та кількість векторів.

Див. визначення в файлі lab3.py, рядок 14

4.2 Простір імен lab3_analysis

Лабораторна робота 3 - Аналіз

Функції

- np.ndarray calc (X, int t=0)

  Обчислює похідну системи по її значенням.
- None analyze ()

  Аналіз системи.

4.2.1 Детальний опис

Лабораторна робота 3 - Аналіз

Результат роботи програми:
```
Якобіан:
Matrix([[2*x1(t) - 2*x2(t), -2*x1(t) + 2*x2(t)], [8*x1(t) + x2(t), x1(t) + 8*x2(t)]])
Особливі точки:
[(-2, 1), (-1, 2), (1, -2), (2, -1)]
Точка: -2 1
Власні числа: [-7.3484692283495345j, 7.3484692283495345j]
-7.3484692283495345j  - центр
7.3484692283495345j  - центр
Точка: -1 2
Власні числа: [(-4.116843969807043+0j), (13.116843969807043+0j)]
(-4.116843969807043+0j)  - сідло
(13.116843969807043+0j)  - сідло
Точка: 1 -2
Власні числа: [(-13.116843969807043+0j), (4.116843969807043+0j)]
(-13.116843969807043+0j)  - сідло
(4.116843969807043+0j)  - сідло
Точка: 2 -1
Власні числа: [-7.3484692283495345j, 7.3484692283495345j]
-7.3484692283495345j  - центр
7.3484692283495345j  - центр
```

4.2.2 Опис функцій

4.2.2.1 analyze() None lab3_analysis.analyze ( )

Аналіз системи.

Див. визначення в файлі lab3_analysis.py, рядок 51
```
00051 def analyze() -> None:
00052     t = symbols("t")
00053     x = symbols("x1 x2", cls=Function)
00054     x = [i(t) for i in x]
00055     f = calc(x, t)
00056 
00057     I = Matrix([[i.diff(j) for j in x] for i in f])
00058     print("Якобіан:")
00059     print(I)
00060 
00061     solutions = solve(f, x)
00062     print("Особливі точки:")
00063     print(solutions)
00064 
00065     for X, Y in solutions:
00066         print()
00067         print("Точка:", X, Y)
00068         eigenvals = list(I.subs(x[0], X).subs(x[1], Y).eigenvals().keys())
00069         eigenvals = [complex(i) for i in eigenvals]
00070         print("Власні числа:", eigenvals)
00071         for i in eigenvals:
00072             if i.imag == 0:
00073                 print(i, " - сідло")
00074             elif i.imag != 0:
00075                 if i.real == 0:
00076                     print(i, " - центр")
00077 
00078 analyze()
```

4.2.2.2 calc()

```
np.ndarray lab3_analysis.calc (
    X,
    int  t = 0 )
```

Обчислює похідну системи по її значенням.

$$\{ x_1'(t) = x_1(t)^2 - 2x_1(t)x_2(t) + x2(t)^2 - 9x_2'(t) = 4x_1(t)^2 + x_1x_2(t) + 4x_2(t)^2 - 18$$

Аргументи

X	точка системи
t	час

Повертає

вектор похідної

Див. визначення в файлі lab3_analysis.py, рядок 43

```
00043 def calc(X, t: int = 0) -> np.ndarray:
00044     x1, x2 = X[0], X[1]
00045     return np.array([
00046         x1**2-2*x1*x2+x2**2-9,
00047         4*x1**2+x1*x2+4*x2**2-18,
00048     ])
00049
```

4.3 Простір імен lab4

Лабораторна робота 4.

Функції

- None f (float x, float a)

  Відображення з постановки задачі.
- None draw_diagram ()

  Друкує діаграму.

Змінні

- int firstIters = 1024

  Скільки перших ітерацій пропустити
- int lastIters = 256

  Скільки останніх ітерацій відобразити на діаграмі
- int aIters = 2000

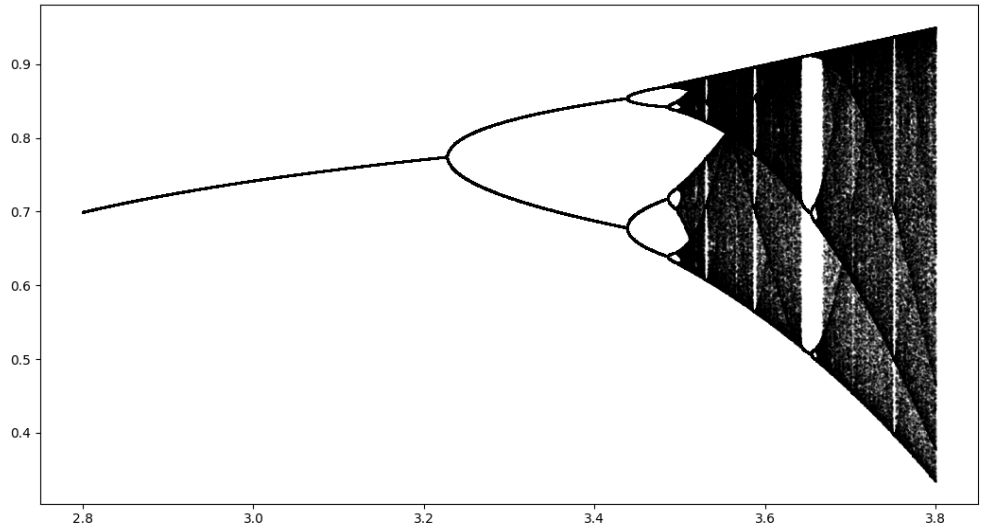
  Скільки значень α побудувати для діаграми.
- list alphaRange = [2.8, 3.8]

  Діапазон значень α [2.8, 3.8].
- float xStart = 0.5

  Стартове значення x_0.

4.3.1 Детальний опис

Лабораторна робота 4.



4.3.2 Опис функцій

4.3.2.1 draw_diagram() None lab4.draw_diagram ( )

Друкує діаграму.

Див. визначення в файлі lab4.py, рядок 32

```
00032 def draw_diagram() -> None:
00033     # Знаходимо усі можливі \f$\alpha\f$
00034     As = np.linspace(alphaRange[0], alphaRange[1], aIters)
00035     # Для кожної \f$\alpha\f$ зберегти останні ітерації
00036     grid = np.zeros((lastIters, aIters))
00037 
00038     # Робимо ітерації для кожної \f$\alpha\f$
00039     for i in range(aIters):
00040         x = xStart
00041         a = As[i]
00042         # Пропускаємо перші ‘firstIters‘ ітерації.
00043         for _ in range(firstIters):
00044             x = f(x, a)
00045 
00046         # Зберінаємо останні ‘lastIters‘ ітерацій.
00047         for j in range(lastIters):
00048             x = f(x, a)
00049             grid[j, i] = x
00050 
00051     # Друкуємо графік.
00052     for i in grid:
00053         plt.scatter(As, i, 1, alpha=0.25, color="black")
00054     plt.show()
00055 
00056 
00057 draw_diagram()
```

4.3.2.2 f()

```
None lab4.f (
    float x,
    float a )
```

Відображення з постановки задачі.

$$x_{n+1} = \alpha x_n^2(1 - x_n^2),\ n = 0, 1, 2, \ldots$$

Аргументи

x	значення x_n
a	значення α

Див. визначення в файлі lab4.py, рядок 28

```
00028 def f(x: float, a: float) -> None:
00029     return a*x**2*(1-x**2)
00030
```

4.3.3 Опис змінних

4.3.3.1 aIters

int lab4.aIters = 2000

Скільки значень α побудувати для діаграми.

Див. визначення в файлі lab4.py, рядок 15

4.3.3.2 alphaRange

list lab4.alphaRange = [2.8, 3.8]

Діапазон значень α [2.8, 3.8].

Див. визначення в файлі lab4.py, рядок 17

4.3.3.3 firstIters

int lab4.firstIters = 1024

Скільки перших ітерацій пропустити

Див. визначення в файлі lab4.py, рядок 11

4.3.3.4 lastIters

int lab4.lastIters = 256

Скільки останніх ітерацій відобразити на діаграмі

Див. визначення в файлі lab4.py, рядок 13

4.3.3.5 xStart float lab4.xStart = 0.5

Стартове значення x_0.

Яке обчислено за допомогою рівняння:

$$f'(x_0, \alpha) = 0$$

Див. визначення в файлі lab4.py, рядок 21

4.4 Простір імен lab5

Лабораторна робота 5.

Функції

- np.ndarray sys (X, t=0)

  Обчислює похідну для системи.
- None draw_attractor ()

  Побудувати аттрактор

Змінні

- int tEnd = 10

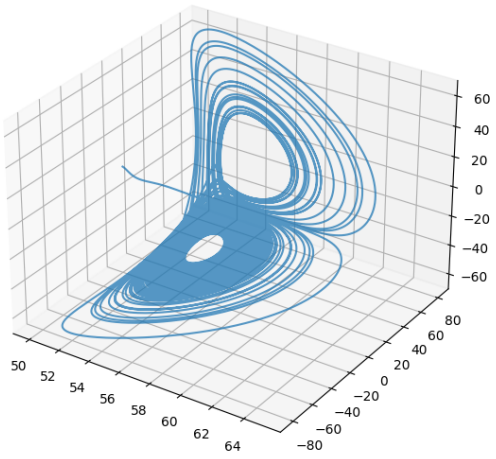
  Час на якому траекторія зупиниться.
- int tSize = 100000

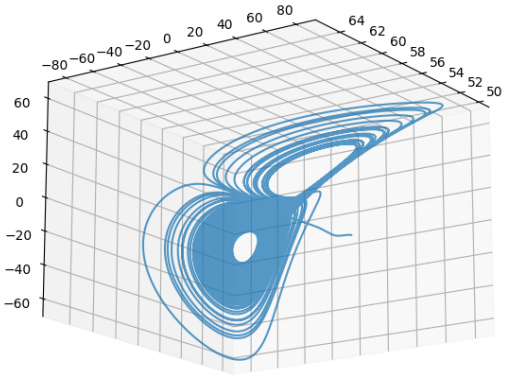
  Кількість часових точок.
- list startPoint = [50, 2, 2]

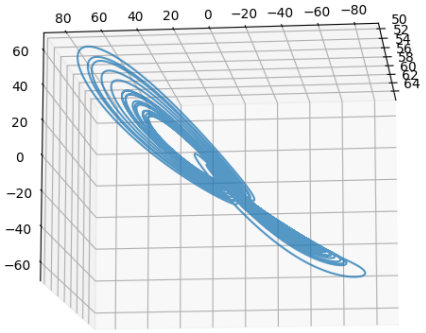
  Початкова точка

4.4.1 Детальний опис

Лабораторна робота 5.







4.4.2 Опис функцій

4.4.2.1 draw_attractor() None lab5.draw_attractor ( )

Побудувати аттрактор

Див. визначення в файлі lab5.py, рядок 41

```
00041 def draw_attractor() -> None:
00042     tspan = np.linspace(0, tEnd, tSize)
00043     ys = odeint(sys, startPoint, tspan)
00044     fig = plt.figure()
00045     ax = plt.axes(projection='3d')
```

```
00046     ax.plot3D(ys[:,0], ys[:,1], ys[:,2], alpha=0.75)
00047     plt.show()
00048
00049
00050 draw_attractor()
```

4.4.2.2 sys() np.ndarray lab5.sys (X, t = 0)

Обчислює похідну для системи.

$$\{ x'(t) = 3x(t) - y(t)^2 + z^2(t) y'(t) = -y(t) - 700z(t) + 10x(t)y(t) z'(t) = x(t) + 700y(t) - z(t) - 15x(t)z(t)$$

Аргументи

X	точка системи
t	час

Повертає

вектор похідної

Див. визначення в файлі lab5.py, рядок 31

```
00031 def sys(X, t=0) -> np.ndarray:
00032     x, y, z = X
00033     return np.array([
00034         3*x-y**2+z**2,
00035         -y-700*z+10*x*y,
00036         x+700*y-z-15*x*z
00037     ])
00038
00039
```

4.4.3 Опис змінних

4.4.3.1 startPoint list lab5.startPoint = [50, 2, 2]

Початкова точка

Див. визначення в файлі lab5.py, рядок 19

4.4.3.2 tEnd int lab5.tEnd = 10

Час на якому траєкторія зупиниться.

Див. визначення в файлі lab5.py, рядок 15

4.4.3.3 tSize int lab5.tSize = 100000

Кількість часових точок.

Див. визначення в файлі lab5.py, рядок 17

5 Файли

5.1 Файл lab3.py

Простори імен

- lab3

  Лабораторна робота 3.

Функції

- np.ndarray lab3.calc (X, int t=0)

  Обчислює похідну системи по її значенням.
- None lab3.print_vector_field (np.ndarray x, np.ndarray y)

  Обчислює векторне поле у діапазонах [-3, 3] по обом осям.
- None lab3.print_trajectories (list starts)

  Будуємо траєкторії.

Змінні

- lab3.x = np.linspace(-3.0, 3.0, 20)

  Задаємо діапазон x для побудови векторного поля та кількість векторів.
- lab3.y = np.linspace(-3.0, 3.0, 20)

  Задаємо діапазон y для побудови векторного поля та кількість векторів.
- list lab3.starts

  Задаємо декілька траєкторій.

5.2 lab3.py

```
00001 #!/usr/bin/env python3
00002
00006
00007 import matplotlib.pyplot as plt
00008 import numpy as np
00009 from scipy.integrate import odeint
00010
00011
00012 x = np.linspace(-3.0, 3.0, 20)
00013
00014 y = np.linspace(-3.0, 3.0, 20)
00015
00024 starts = [
00025     ([0, 0], 0.6),
00026     ([2, -1.585], 100),
00027     ([-2.0, 1.0], 100),
00028     ([1, -2], 100),
00029     ([-1, 2], 100),
00030 ]
00031
00032
00039 def calc(X, t: int = 0) -> np.ndarray:
00040     x1, x2 = X[0], X[1]
```

```
00041     return np.array([
00042         x1**2-2*x1*x2+x2**2-9,
00043         4*x1**2+x1*x2+4*x2**2-18,
00044     ])
00045
00046
00049 def print_vector_field(x: np.ndarray, y: np.ndarray) -> None:
00050     X, Y = np.meshgrid(x, y)
00051     u, v = np.zeros(X.shape), np.zeros(Y.shape)
00052     NI, NJ = X.shape
00053     # Обчислює векторне поле по заданим значенням.
00054     for i in range(NI):
00055         for j in range(NJ):
00056             u[i, j], v[i, j] = calc([X[i, j], Y[i, j]])
00057     # Друкує на графік векторне поле.
00058     plt.quiver(X, Y, u, v, color='r')
00059
00060
00062 def print_trajectories(starts: list) -> None:
00063     # Обчислюємо траєкторії задані у змінній `starts`
00064     for s, t in starts:
00065         # Задаємо діапазон часу [0, t] та ділить відрізок на 2000 частин.
00066         tspan = np.linspace(0, t, 2000)
00067         # Інтегрує систему до заданого часу `t` та початкових значень `s`
00068         tr = odeint(calc, s, tspan)
00069         # Друкуєму саму траєкторію
00070         plt.plot(tr[:,0], tr[:,1])
00071         # Друкуємо початок траєкторії у вигляді кружечка
00072         plt.plot([tr[0,0]], [tr[0,1]], 'o')
00073         # Друкуємо кінець траєкторії у вигляді квадратика
00074         plt.plot([tr[-1,0]], [tr[-1,1]], 's')
00075
00076
00077 print_vector_field(x, y)
00078 print_trajectories(starts)
00079 # Друкуємо графік
00080 plt.show()
```

5.3 Файл lab3_analysis.py

Простори імен

- lab3_analysis

  Лабораторна робота 3 - Аналіз

Функції

- np.ndarray lab3_analysis.calc (X, int t=0)

  Обчислює похідну системи по її значенням.
- None lab3_analysis.analyze ()

  Аналіз системи.

5.4 lab3_analysis.py

```
00001 #!/usr/bin/env python3
00002
00032
00033 from sympy import *
00034 import numpy as np
00035
00036
00043 def calc(X, t: int = 0) -> np.ndarray:
00044     x1, x2 = X[0], X[1]
00045     return np.array([
00046         x1**2-2*x1*x2+x2**2-9,
00047         4*x1**2+x1*x2+4*x2**2-18,
00048     ])
00049
00050
00051 def analyze() -> None:
00052     t = symbols("t")
00053     x = symbols("x1 x2", cls=Function)
```

```
00054     x = [i(t) for i in x]
00055     f = calc(x, t)
00056
00057     I = Matrix([[i.diff(j) for j in x] for i in f])
00058     print("Якобіан:")
00059     print(I)
00060
00061     solutions = solve(f, x)
00062     print("Особливі точки:")
00063     print(solutions)
00064
00065     for X, Y in solutions:
00066         print()
00067         print("Точка:", X, Y)
00068         eigenvals = list(I.subs(x[0], X).subs(x[1], Y).eigenvals().keys())
00069         eigenvals = [complex(i) for i in eigenvals]
00070         print("Власні числа:", eigenvals)
00071         for i in eigenvals:
00072             if i.imag == 0:
00073                 print(i, " - сідло")
00074             elif i.imag != 0:
00075                 if i.real == 0:
00076                     print(i, " - центр")
00077
00078 analyze()
```

5.5 Файл lab4.py

Простори імен

- lab4

  Лабораторна робота 4.

Функції

- None lab4.f (float x, float a)

  Відображення з постановки задачі.
- None lab4.draw_diagram ()

  Друкує діаграму.

Змінні

- int lab4.firstIters = 1024

  Скільки перших ітерацій пропустити
- int lab4.lastIters = 256

  Скільки останніх ітерацій відобразити на діаграмі
- int lab4.aIters = 2000

  Скільки значень α побудувати для діаграми.
- list lab4.alphaRange = [2.8, 3.8]

  Діапазон значень α [2.8, 3.8].
- float lab4.xStart = 0.5

  Стартове значення x_0.

5.6 lab4.py

```
00001 #!/usr/bin/env python3
00002
00006
00007 import matplotlib.pyplot as plt
00008 import numpy as np
00009
00010
00011 firstIters = 1024
00012
00013 lastIters = 256
00014
00015 aIters = 2000
00016
00017 alphaRange = [2.8, 3.8]
00018
00021 xStart = 0.5
00022
00023
00024
00028 def f(x: float, a: float) -> None:
00029     return a*x**2*(1-x**2)
00030
00031
00032 def draw_diagram() -> None:
00033     # Знаходимо усі можливі \f$\alpha\f$
00034     As = np.linspace(alphaRange[0], alphaRange[1], aIters)
00035     # Для кожної \f$\alpha\f$ зберегти останні ітерації
00036     grid = np.zeros((lastIters, aIters))
00037
00038     # Робимо ітерації для кожної \f$\alpha\f$
00039     for i in range(aIters):
00040         x = xStart
00041         a = As[i]
00042         # Пропускаємо перші ‘firstIters‘ ітерації.
00043         for _ in range(firstIters):
00044             x = f(x, a)
00045
00046         # Зберінаємо останні ‘lastIters‘ ітерацій.
00047         for j in range(lastIters):
00048             x = f(x, a)
00049             grid[j, i] = x
00050
00051     # Друкуємо графік.
00052     for i in grid:
00053         plt.scatter(As, i, 1, alpha=0.25, color="black")
00054     plt.show()
00055
00056
00057 draw_diagram()
```

5.7 Файл lab5.py

Простори імен

- lab5

  Лабораторна робота 5.

Функції

- np.ndarray lab5.sys (X, t=0)

  Обчислює похідну для системи.
- None lab5.draw_attractor ()

  Побудувати аттрактор

Змінні

- int lab5.tEnd = 10

  Час на якому траекторія зупиниться.
- int lab5.tSize = 100000

  Кількість часових точок.
- list lab5.startPoint = [50, 2, 2]

  Початкова точка

5.8 lab5.py

```
00001 #!/usr/bin/env python3
00002
00010 import matplotlib.pyplot as plt
00011 import numpy as np
00012 from scipy.integrate import odeint
00013
00014
00015 tEnd = 10
00016
00017 tSize = 100000
00018
00019 startPoint = [50, 2, 2]
00020
00021
00022
00031 def sys(X, t=0) -> np.ndarray:
00032     x, y, z = X
00033     return np.array([
00034         3*x-y**2+z**2,
00035         -y-700*z+10*x*y,
00036         x+700*y-z-15*x*z
00037     ])
00038
00039
00040
00041 def draw_attractor() -> None:
00042     tspan = np.linspace(0, tEnd, tSize)
00043     ys = odeint(sys, startPoint, tspan)
00044     fig = plt.figure()
00045     ax = plt.axes(projection='3d')
00046     ax.plot3D(ys[:,0], ys[:,1], ys[:,2], alpha=0.75)
00047     plt.show()
00048
00049
00050 draw_attractor()
```

5.9 Файл mainpage.dox