Лабораторні роботи 1-3

Створено системою Doxygen 1.9.2

1 Звіт з лабораторних робіт №1-3

за дисципліною "Big Data Application and Analytics"
студента групи ПК-21m-1
Панасенка Єгора Сергійовича

Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2021-2022 навч.р.
Варіант 13

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/BDAnA_Labs/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/09/BDAnA_Labs

1.1 Постановка задачі

Комплекс лабораторних робіт №13.

СЕМЕСТР 2

I. Для наступної системи дослідити стійкість положення рівноваги:

$$\begin{cases} \dot{x}_1(t) = -x_1(t) + 2x_2(t) \\ \dot{x}_2(t) = -2x_2(t) \\ \dot{x}_3(t) = -2x_1(t) - 2x_2(t) - x_3(t). \end{cases}$$

II. Побудувати бифуркаційну діаграму для наступного дискретного відображення і вказати значення параметра α, при якому виникає перехід до хаосу (перед тим встановити відрізок I числової вісі, інваріантний відносно заданого відображення $f(x, \alpha)$: $f(I, \alpha) \subset I$):

$$x_{n+1} = \alpha + x_n^2 + x_n, n = 0, 1, 2, ...$$

Побудувати рекурентну діаграму для часового ряду, що генерується відображенням $f(x, \alpha)$, при хаотичному значенні параметра $\alpha = \alpha^*$.

III.Побудувати хаотичний атрактор для наступної 3-вимірної системи автономних диференціальних рівнянь:

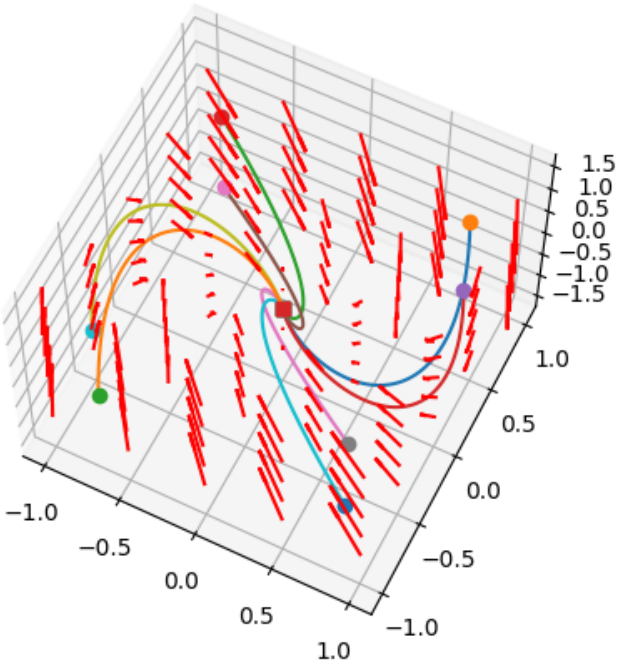
$$\begin{cases} \dot{x}(t) = 3x(t) + x(t)(x(t) - 3)(5y^2(t) - z^2(t))/(1 + y^2(t) + z^2(t)), \\ \dot{y}(t) = 2y(t) - 14z(t) - 5(x(t) - 3)y(t), \\ \dot{z}(t) = 14y(t) + 2z(t) + 5(x(t) - 3)z(t). \end{cases}$$

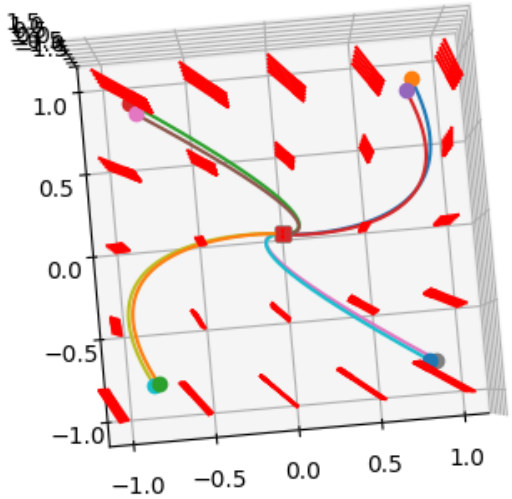
Зробити повний рекурентнмй аналіз для часового ряду, що генерується змінною $x(t)$, $y(t)$ або $z(t)$ (винайти вимірність простору вкладення, поріг, запізнення та побудувати рекурентну діаграму).

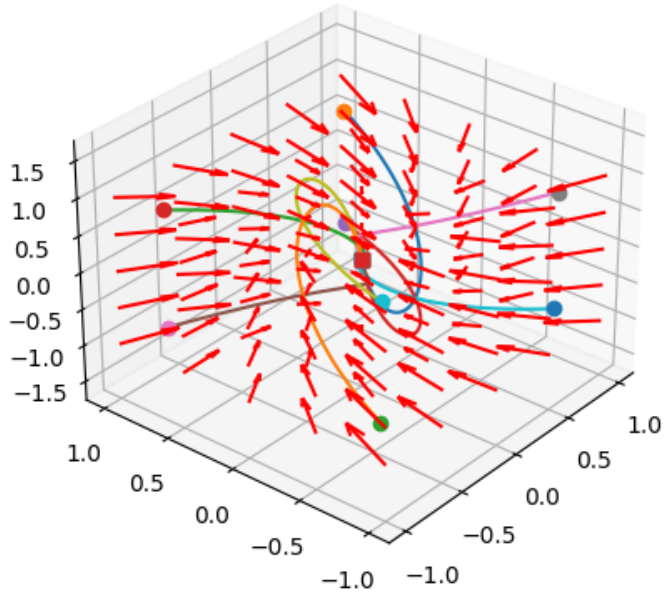
1.2 Отримані результати

1.2.1 Лабораторна робота 1

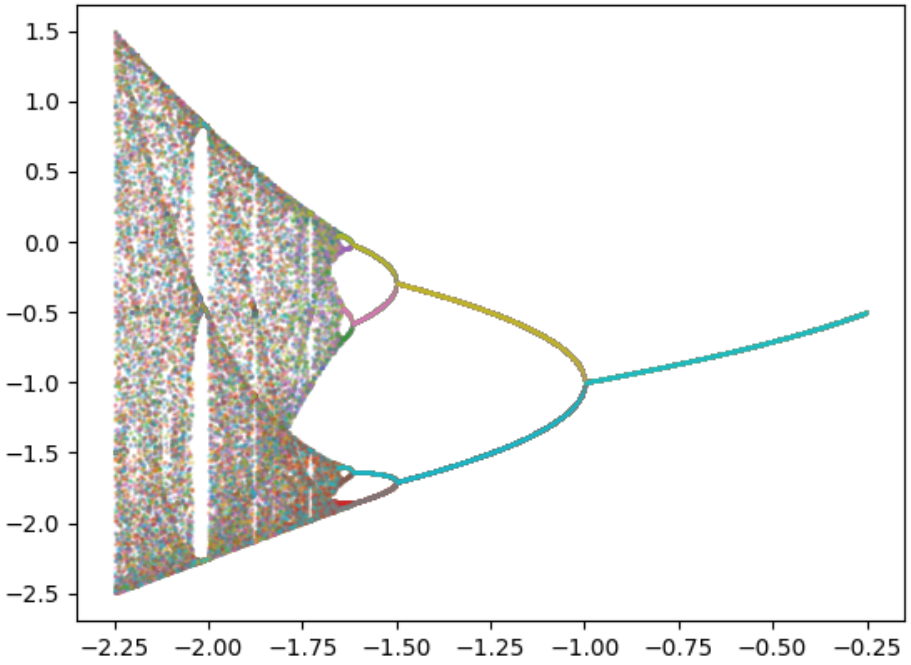
```
[     -x1(t) + 2*x2(t)     ]
[                          ]
[         -2*x2(t)         ]
[                          ]
[-2*x1(t) - 2*x2(t) - x3(t)]
Якобіан:
[-1   2    0 ]
[            ]
[0    -2   0 ]
[            ]
[-2   -2   -1]
Особливі точки:
[[0, 0, 0]]
Точка: 0 0 0
{-2: 1, -1: 2}
Власні числа: [(-2+0j), (-1+0j)]
Стійкий вузол
```


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

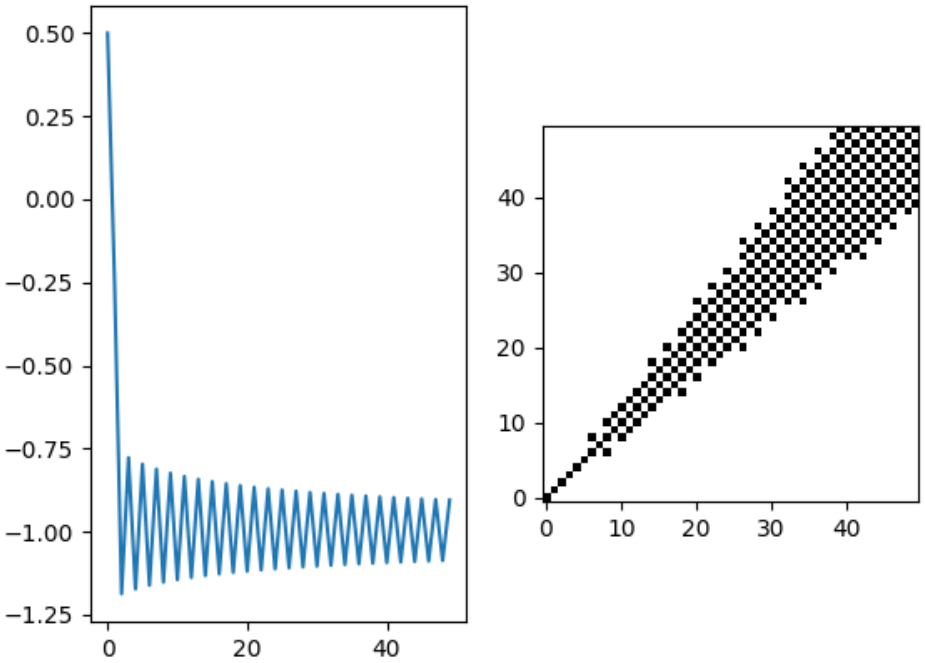

1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0



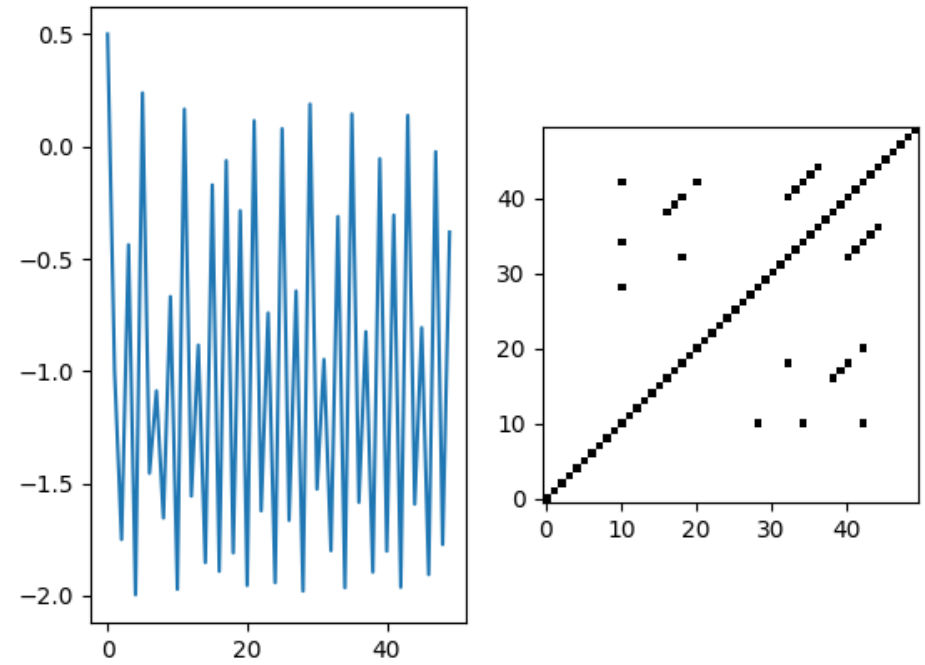
1.2.2 Лабораторна робота 2

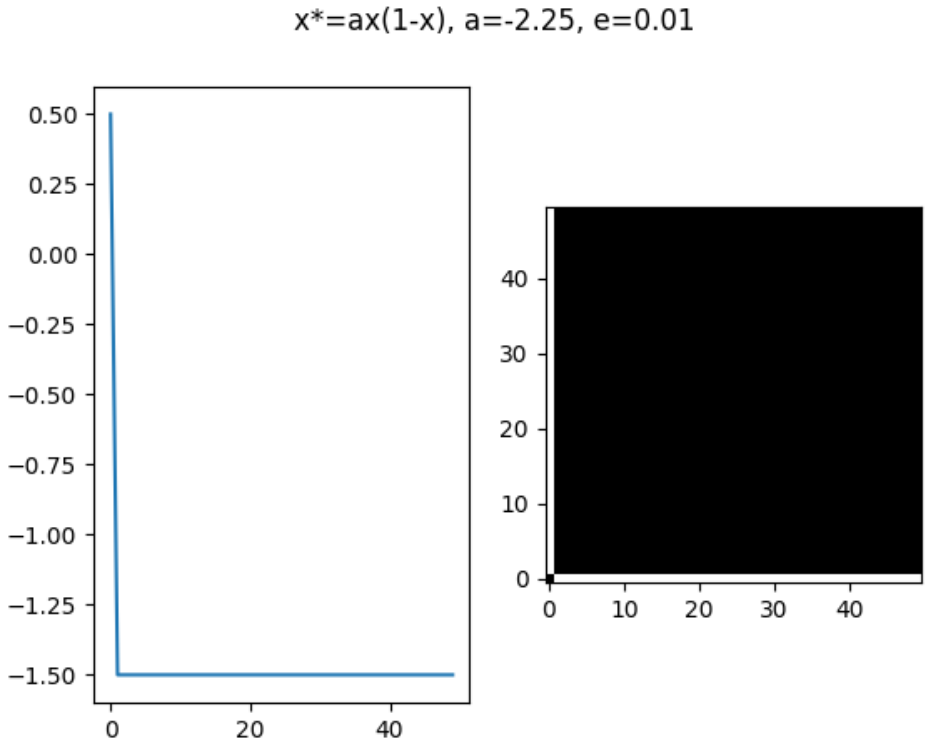


x*=ax(1-x), a=-1, e=0.01

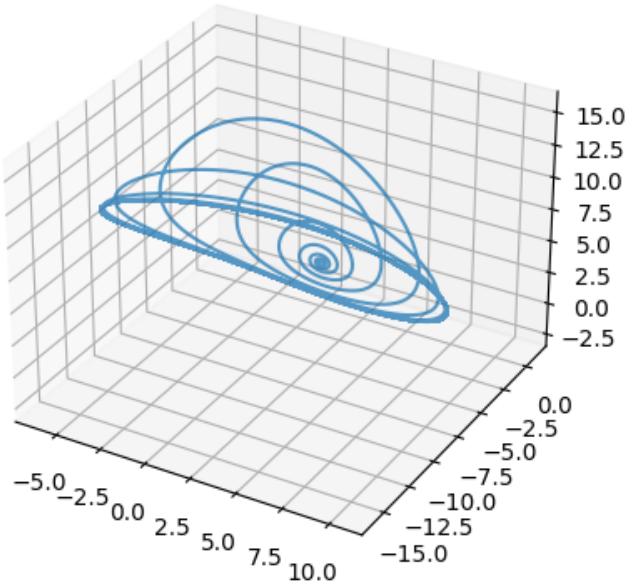


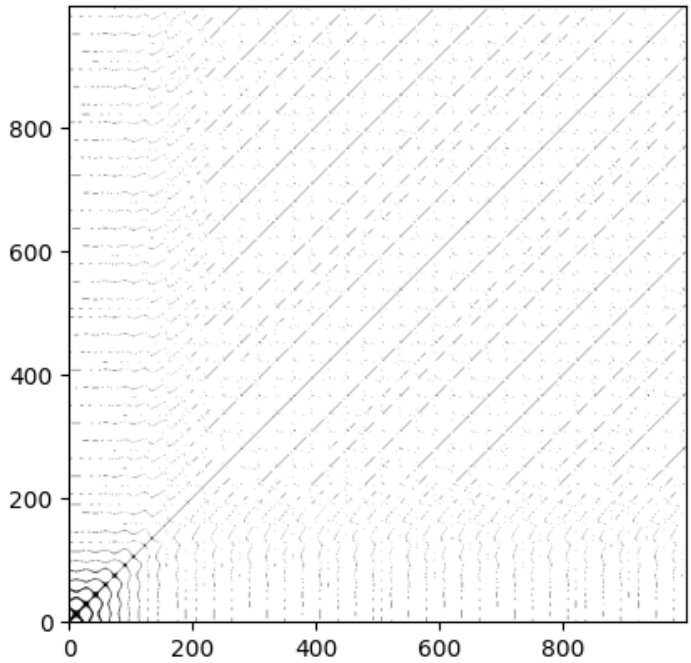
x*=ax(1-x), a=-1.75, e=0.01

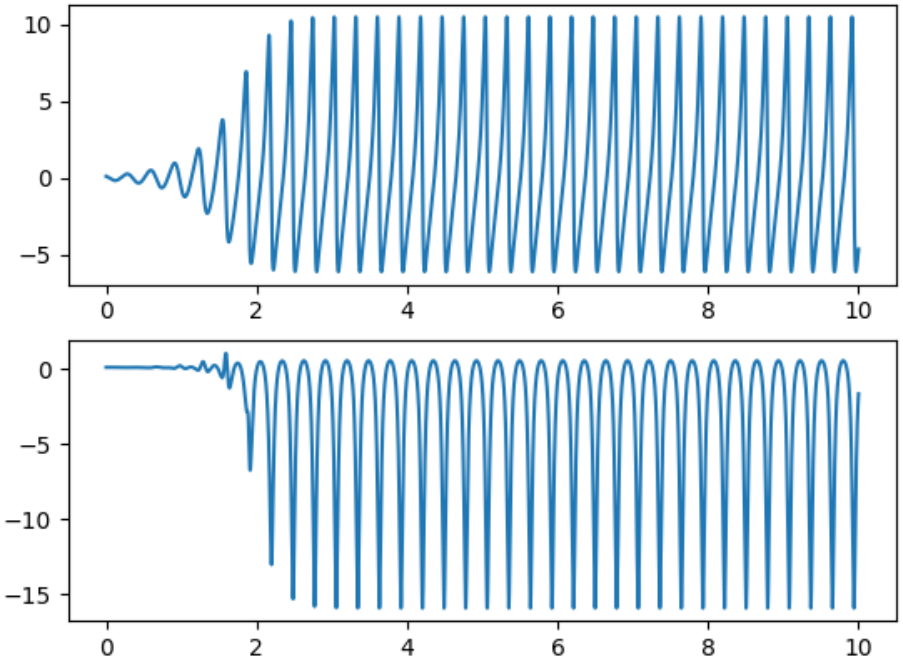


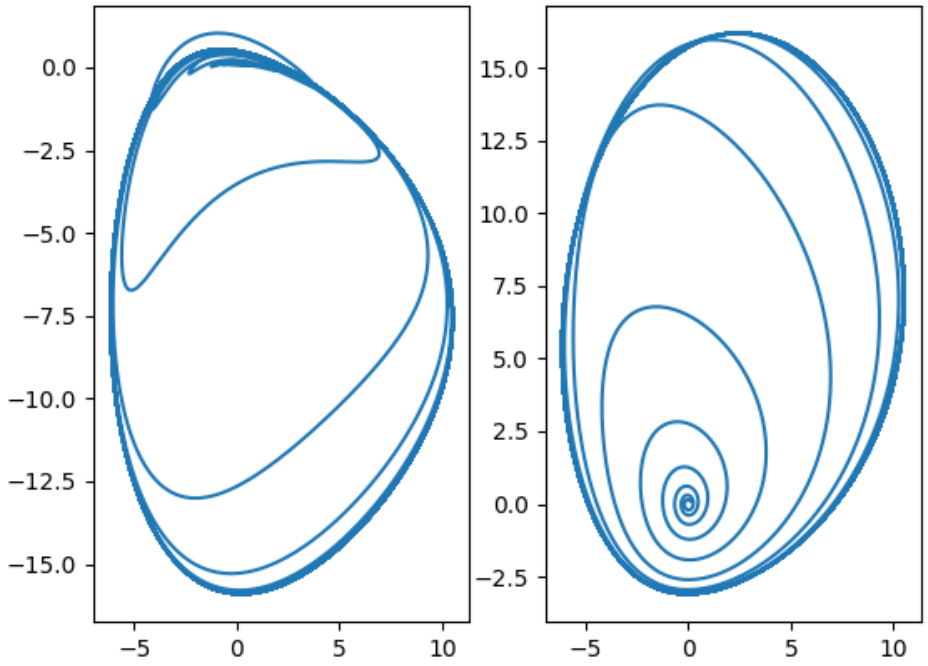


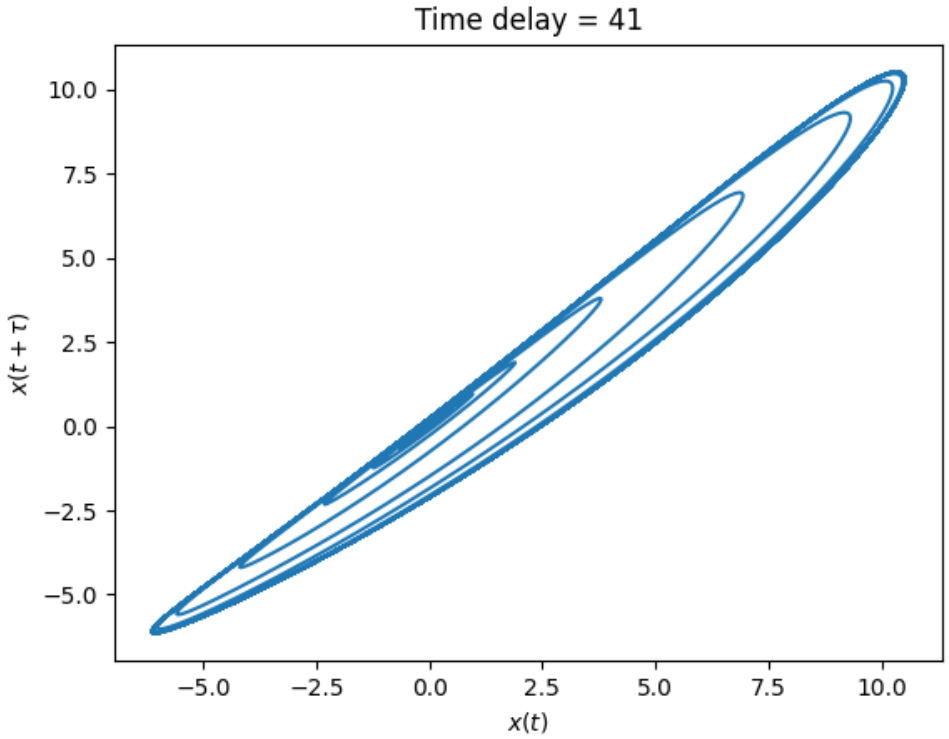
1.2.3 Лабораторна робота 3

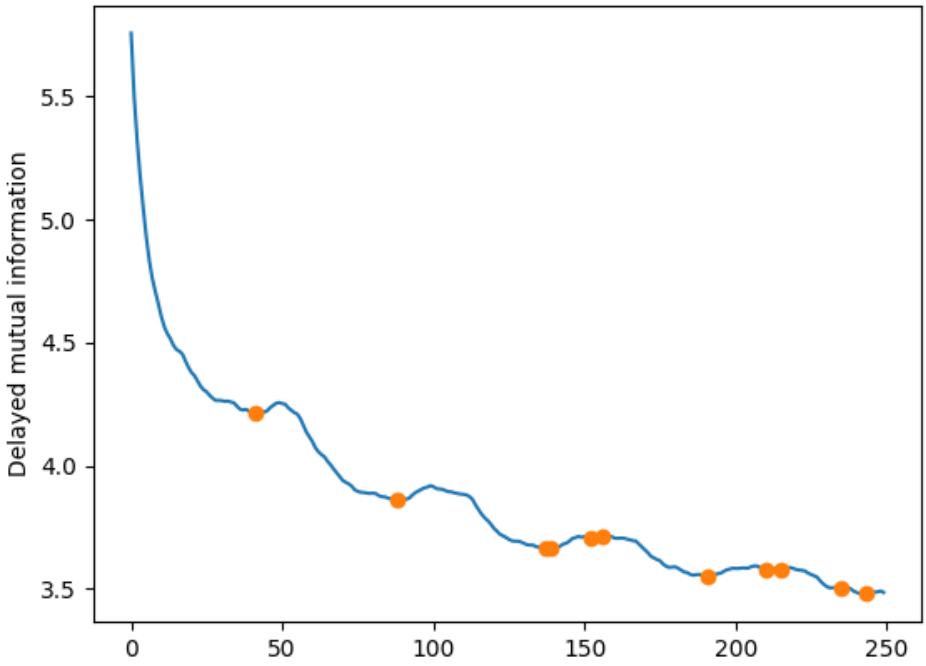


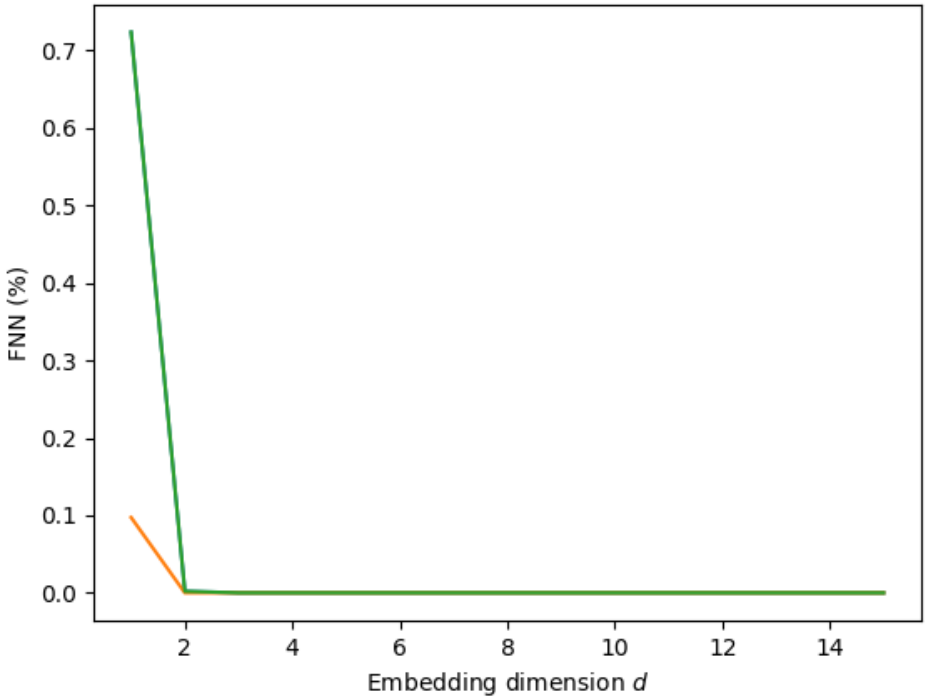

800
600
400
200
0
0
200
400
600
800

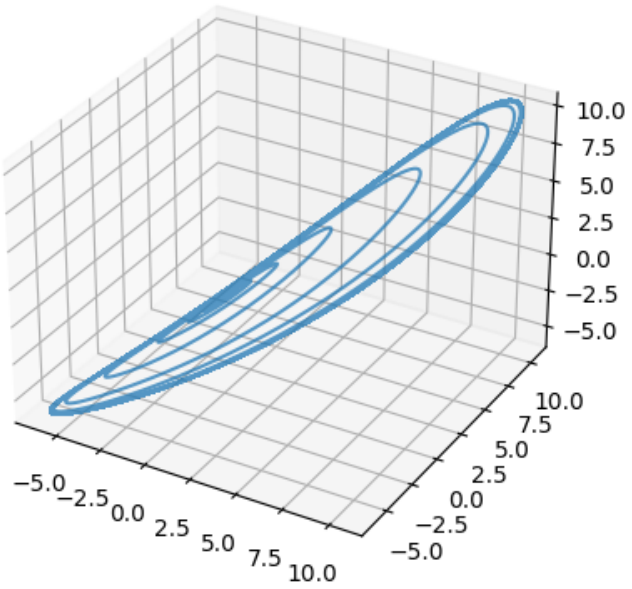

10
5
0
−5
0
2
4
6
8
10
0
−5
−10
−15
0
2
4
6
8
10


0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
−5
0
5
10
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5
0
5
10


Time delay = 41
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
x(t + τ)
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x(t)


Delayed mutual information
5.5
5.0
4.5
4.0
3.5
0
50
100
150
200
250


FNN (%)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2
4
6
8
10
12
14
Embedding dimension d



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 BDAnA_Lab1_13

Функції

- np.ndarray calc (X, int t=0)
- None print_vector_field (ax, x, y, z)
- None print_trajectories (ax, list starts)

  Будуємо траєкторії.

Змінні

- x = np.linspace(-1.0, 1.0, 5)
- y = np.linspace(-1.0, 1.0, 5)
- z = np.linspace(-1.0, 1.0, 5)
- list starts
- fig = plt.figure()
- ax = fig.gca(projection='3d')

4.1.1 Опис функцій

4.1.1.1 calc() np.ndarray BDAnA_Lab1_13.calc (X, int t = 0)

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 24

```
00024 def calc(X, t: int = 0) -> np.ndarray:
00025     x1, x2, x3 = X[0], X[1], X[2]
00026     return np.array([
00027         -x1+2*x2, -2*x2, -2*x1-2*x2-x3
00028     ])
00029
00030
```

4.1.1.2 print_trajectories() None BDAnA_Lab1_13.print_trajectories (ax, list starts)

Будуємо траєкторії.

Аргументи

starts	змінна starts, яка задає усі траєкторії

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 41

```
00041 def print_trajectories(ax, starts: list) -> None:
00042     # Обчислюємо траєкторії задані у змінній 'starts'
```

```
00043     for s, t in starts:
00044         # Задаємо діапазон часу [0, t] та ділить відрізок на 2000 частин.
00045         tspan = np.linspace(0, t, 2000)
00046         # Інтегрує систему до заданого часу ‘t‘ та початкових значень ‘s‘
00047         tr = odeint(calc, s, tspan)
00048         # Друкуєму саму траєкторію
00049         ax.plot3D(tr[:, 0], tr[:, 1], tr[:, 2])
00050         # Друкуємо початок траєкторії у вигляді кружечка
00051         ax.plot3D([tr[0, 0]], [tr[0, 1]], [tr[0, 2]], 'o')
00052         # Друкуємо кінець траєкторії у вигляді квадратика
00053         ax.plot3D([tr[-1, 0]], [tr[-1, 1]], [tr[-1, 2]], 's')
00054
00055
```

4.1.1.3 print_vector_field() None BDAnA_Lab1_13.print_vector_field (ax, x, y, z)

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 31

```
00031 def print_vector_field(ax, x, y, z) -> None:
00032     X, Y, Z = np.meshgrid(x, y, z)
00033     u, v, w = np.zeros(X.shape), np.zeros(Y.shape), np.zeros(Z.shape)
00034     u, v, w = calc([X, Y, Z])
00035     # Друкує на графік векторне поле.
00036     ax.quiver(X, Y, Z, u, v, w, color='r', length=0.1)
00037
00038
```

4.1.2 Опис змінних

4.1.2.1 ax BDAnA_Lab1_13.ax = fig.gca(projection='3d')

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 57

4.1.2.2 fig BDAnA_Lab1_13.fig = plt.figure()

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 56

4.1.2.3 starts list BDAnA_Lab1_13.starts

Початкові значення

```
00001 = [
00002     ([0.8, 0.8, 0.8], 10),
00003     ([0.8, 0.8, -0.8], 10),
00004     ([0.8, -0.8, 0.8], 10),
00005     ([0.8, -0.8, -0.8], 10),
00006     ([-0.8, 0.8, 0.8], 10),
00007     ([-0.8, 0.8, -0.8], 10),
00008     ([-0.8, -0.8, 0.8], 10),
00009     ([-0.8, -0.8, -0.8], 10),
00010 ]
```

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 12

4.1.2.4 x BDAnA_Lab1_13.x = np.linspace(-1.0, 1.0, 5)

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 8

4.1.2.5 y BDAnA_Lab1_13.y = np.linspace(-1.0, 1.0, 5)

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 9

4.1.2.6 z BDAnA_Lab1_13.z = np.linspace(-1.0, 1.0, 5)

Див. визначення в файлі BDAnA_Lab1_13.py, рядок 10

4.2 Простір імен BDAnA_Lab1_13_Analysis

Функції

- np.ndarray calc (X, int t=0)
- None analyze ()

  Аналіз системи.

Змінні

- use_unicode

4.2.1 Опис функцій

4.2.1.1 analyze() None BDAnA_Lab1_13_Analysis.analyze ( )

Аналіз системи.

Див. визначення в файлі BDAnA_Lab1_13_Analysis.py, рядок 14

```
00014 def analyze() -> None:
00015     t = symbols("t")
00016     x = symbols("x1 x2 x3", cls=Function)
00017     x = [i(t) for i in x]
00018     f = calc(x, t)
00019 
00020     # ~ I = Matrix([[i.diff(j) for j in x] for i in f])
00021     pprint(Matrix(f))
00022     I = Matrix(f).jacobian(x)
00023     pprint("Якобіан:")
00024     pprint(I)
00025 
00026     solutions = [list(i.values()) for i in solve(f)]
00027     pprint("Особливі точки:")
00028     pprint(solutions)
00029 
00030     for X, Y, Z in solutions:
00031         print()
00032         print("Точка:", X, Y, Z)
00033         pprint(lambdify(x, I, modules="sympy")(X, Y, Z).eigenvals())
00034         eigenvals = list(lambdify(x, I, modules="sympy")(X, Y, Z).eigenvals().keys())
```

```
00035         eigenvals = [complex(i) for i in eigenvals]
00036         print("Власні числа:", eigenvals)
00037         if all([i.imag != 0 for i in eigenvals]):
00038             if all([i.real == 0 for i in eigenvals]):
00039                 print("Центр")
00040                 print("Мабуть я у планетарії...")
00041             elif all([i.real < 0 for i in eigenvals]):
00042                 print("Стійкий фокус")
00043                 print("Голова паморочиться... Але рівновагу не втрачаю і голова на місці.")
00044             elif all([i.real > 0 for i in eigenvals]):
00045                 print("Нестійкий фокус")
00046                 print("Голова паморочиться... Зовсім рівновагу втрачаю! Ловіть мене!")
00047             else:
00048                 print("Щось якось обертається, але не знаю як. Розповіси як зустрінимося.")
00049         elif all([i.imag == 0 for i in eigenvals]):
00050             if all([i.real < 0 for i in eigenvals]):
00051                 print("Стійкий вузол")
00052                 print("Очі розлазяться, але десь на одній лінії зустрічаються.")
00053             elif all([i.real > 0 for i in eigenvals]):
00054                 print("Нестійкий вузол")
00055                 print("Очі розлазяться у різні боки.")
00056             else:
00057                 print("Сідло")
00058                 print("Звичайне кінське сідло. Наскільки комфортне не пробував.")
00059         else:
00060             print("А що, так можна було? Комплексні числа у парі зазвичай.")
00061             print("Негайно передзвони та розповіси про такий випадок!")
00062
00063 analyze()
```

4.2.1.2 calc() np.ndarray BDAnA_Lab1_13_Analysis.calc (X, int t = 0)

Див. визначення в файлі BDAnA_Lab1_13_Analysis.py, рядок 7

```
00007 def calc(X, t: int = 0) -> np.ndarray:
00008     x1, x2, x3 = X[0], X[1], X[2]
00009     return np.array([
00010         -x1+2*x2, -2*x2, -2*x1-2*x2-x3
00011     ])
00012
```

4.2.2 Опис змінних

4.2.2.1 use_unicode BDAnA_Lab1_13_Analysis.use_unicode

Див. визначення в файлі BDAnA_Lab1_13_Analysis.py, рядок 5

4.3 Простір імен BDAnA_Lab2_13

Функції

- None f (float x, float a)

  Відображення з постановки задачі.
- None draw_diagram ()

  Друкує діаграму.
- None draw_diagram2 (a, epsillon)

Змінні

- int firstIters = 1024

  Скільки перших ітерацій пропустити
- int lastIters = 50

  Скільки останніх ітерацій відобразити на діаграмі
- int aIters = 2000

  Скільки значень α побудувати для діаграми.
- list alphaRange = [-2.25, 0.25]

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

  Стартове значення x_0.

4.3.1 Опис функцій

4.3.1.1 draw_diagram() None BDAnA_Lab2_13.draw_diagram ( )

Друкує діаграму.

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 27

```
00027 def draw_diagram() -> None:
00028     # Знаходимо усі можливі \f$\alpha\f$
00029     As = np.linspace(alphaRange[0], alphaRange[1], aIters)
00030     # Для кожної \f$\alpha\f$ зберегти останні ітерації
00031     grid = np.zeros((lastIters, aIters))
00032 
00033     # Робимо ітерації для кожної \f$\alpha\f$
00034     for i in range(aIters):
00035         x = xStart
00036         a = As[i]
00037         # Пропускаємо перші ‘firstIters‘ ітерації.
00038         for _ in range(firstIters):
00039             x = f(x, a)
00040 
00041         # Зберінаємо останні ‘lastIters‘ ітерацій.
00042         for j in range(lastIters):
00043             x = f(x, a)
00044             grid[j, i] = x
00045 
00046     # Друкуємо графік.
00047     for i in grid:
00048         plt.scatter(As, i, 1, alpha=0.25)
00049     plt.show(block=False)
00050 
00051 
```

4.3.1.2 draw_diagram2() None BDAnA_Lab2_13.draw_diagram2 (a, epsillon)

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 52

```
00052 def draw_diagram2(a, epsillon) -> None:
00053     x = np.zeros(lastIters)
00054     x[0] = xStart
00055 
00056     for i in range(1, lastIters):
00057         x[i] = f(x[i-1], a)
00058 
00059     xx, yy = np.meshgrid(x, x)
00060     I = (abs(xx - yy) - epsillon) > 0
00061 
```

```
00062     fig, axs = plt.subplots(1,2)
00063     fig.suptitle(f"x*=ax(1-x), a={a}, e={epsillon}")
00064     axs[0].plot(range(lastIters), x)
00065     axs[1].imshow(I, cmap=plt.cm.gray, origin='lower')
00066     plt.show(block=False)
00067
00068
00069 draw_diagram()
00070 draw_diagram2(-1, 0.01)
00071 draw_diagram2(-1.75, 0.01)
00072 draw_diagram2(-2.25, 0.01)
00073 plt.show()
```

4.3.1.3 f()

None BDAnA_Lab2_13.f (
float x,
float a)

Відображення з постановки задачі.

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

Аргументи

x	значення x_n
a	значення α

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 23

```
00023 def f(x: float, a: float) -> None:
00024     return a+x**2+x
00025
```

4.3.2 Опис змінних

4.3.2.1 aIters

int BDAnA_Lab2_13.aIters = 2000

Скільки значень α побудувати для діаграми.

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 10

4.3.2.2 alphaRange

list BDAnA_Lab2_13.alphaRange = [-2.25, 0.25]

Діапазон значень α [2.8, 3.8].

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 12

4.3.2.3 firstIters int BDAnA_Lab2_13.firstIters = 1024

Скільки перших ітерацій пропустити

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 6

4.3.2.4 lastIters int BDAnA_Lab2_13.lastIters = 50

Скільки останніх ітерацій відобразити на діаграмі

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 8

4.3.2.5 xStart float BDAnA_Lab2_13.xStart = 0.5

Стартове значення x_0.

Яке обчислено за допомогою рівняння:

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

Див. визначення в файлі BDAnA_Lab2_13.py, рядок 16

4.4 Простір імен BDAnA_Lab3_13

Функції

- def localmin (x)
- np.ndarray sys (X, t=0)
- None draw_attractor ()

Змінні

- int tEnd = 10
- int tSize = 100000
- int dSize = 1000
- list startPoint = [0.1, 0.1, 0.1]
- float epsillon = 0.1

4.4.1 Опис функцій

4.4.1.1 draw_attractor() None BDAnA_Lab3_13.draw_attractor ( )

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 36

```
00036 def draw_attractor() -> None:
00037     tspan = np.linspace(0, tEnd, dSize)
00038     ys = odeint(sys, startPoint, tspan, rtol=0.0000000001, atol=0.0000000001)
00039     xx, yy = np.meshgrid(ys[:,0], ys[:,0])
00040     I = (abs(xx - yy) - epsillon) > 0
00041 
00042     tspan = np.linspace(0, tEnd, tSize)
00043     yso = ys
00044     ys = odeint(sys, startPoint, tspan)
00045 
00046     lag = np.arange(250)
00047     x = ys[:,0]
00048     r = delay.acorr(x, maxtau=250)
00049     i = delay.dmi(x, maxtau=250)
00050 
00051     i_delay = localmin(noise.sma(i, hwin=1)) + 1
00052     r_delay = np.argmax(r < 1.0 / np.e)
00053 
00054     print(r'Minima of delayed mutual information = %s' % i_delay)
00055     print(r'Autocorrelation time = %d' % r_delay)
00056 
00057     dim = np.arange(1, 15 + 1)
00058     tau_here = (localmin(noise.sma(delay.dmi(x, maxtau=250), hwin=1)) + 1)[0]
00059     tau_here = np.argmax(delay.acorr(yso[:,0], maxtau=250) < 1.0 / np.e)
00060     f = dimension.fnn(yso[:,0], tau=tau_here, dim=dim, window=0, metric='euclidean')
00061 
00062     fig = plt.figure(1)
00063     ax = plt.axes(projection='3d')
00064     ax.plot3D(ys[:,0], ys[:,1], ys[:,2], alpha=0.75)
00065     plt.figure(2)
00066     plt.imshow(I, cmap=plt.cm.gray, origin='lower')
00067     plt.figure(3)
00068     plt.subplot(211)
00069     plt.plot(tspan, ys[:,0])
00070     plt.subplot(212)
00071     plt.plot(tspan, ys[:,1])
00072     plt.figure(4)
00073     plt.subplot(121)
00074     plt.plot(ys[:,0], ys[:,1])
00075     plt.subplot(122)
00076     plt.plot(ys[:,0], ys[:,2])
00077     plt.figure(5)
00078     # ~ plt.subplot(121)
00079     # ~ plt.title(r'Time delay = %d' % r_delay)
00080     # ~ plt.xlabel(r'$x(t)$')
00081     # ~ plt.ylabel(r'$x(t + \tau)$')
00082     # ~ plt.plot(ys[:-r_delay,0], ys[r_delay:,0])
00083     # ~ plt.subplot(122)
00084     plt.title(r'Time delay = %d' % i_delay[0])
00085     plt.xlabel(r'$x(t)$')
00086     plt.ylabel(r'$x(t + \tau)$')
00087     plt.plot(ys[:-i_delay[0],0], ys[i_delay[0]:,0])
00088     plt.figure(6)
00089     plt.ylabel(r'Delayed mutual information')
00090     plt.plot(lag, i, i_delay, i[i_delay], 'o')
00091     plt.figure(7)
00092     plt.plot(dim, f[0], dim, f[1], dim, f[2])
00093     plt.xlabel(r'Embedding dimension $d$')
00094     plt.ylabel(r'FNN (%)')
00095     plt.figure(8)
00096     ax = plt.axes(projection='3d')
00097     ax.plot3D(ys[:-2*i_delay[0],0], ys[i_delay[0]:-i_delay[0],0], ys[2*i_delay[0]:,0], alpha=0.75)
00098     print(ys[-1])
00099     plt.show()
00100 
00101 
00102 draw_attractor()
```

4.4.1.2 localmin() def BDAnA_Lab3_13.localmin (x)

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 14

```
00014 def localmin(x):
00015     return (np.diff(np.sign(np.diff(x))) > 0).nonzero()[0] + 1
00016 
00017 
```

4.4.1.3 sys() np.ndarray BDAnA_Lab3_13.sys (X, t = 0)

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 18

```
00018 def sys(X, t=0) -> np.ndarray:
00019     y1, y2, y3 = X
00020     return np.array([
00021         # ~ 1000-3*x-1000*y**2+10*z**2,
00022         # ~ y+2*z+x*(y+z*4/3),
00023         # ~ -2*y+z+x*(-y*4/3+z)
00024         # ~ 3*x+x*(x-3)*(5*y**2-z**2)/(1+y**2+z**2),
00025         # ~ 2*y-14*z-5*(x-3)*y,
00026         # ~ 14*y+2*z+5*(x-3)*z
00027         # ~ -3*x+140*y**2-z**2,
00028         # ~ y-200*z-140*x*y,
00029         # ~ 200*y+z+x*z
00030         2*y1-20*y3+3*y1**2-2*y2**2-y3**2-2*y2*y3-2*y1*y3,
00031         -0.5*y2+4*y2**2+8*y1*y2+4*y2*y3+4*y1*y3,
00032         20*y1+2*y3+4*y1*y3+2*y2*y3+y3**2
00033     ])
00034
00035
```

4.4.2 Опис змінних

4.4.2.1 dSize int BDAnA_Lab3_13.dSize = 1000

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 9

4.4.2.2 epsillon float BDAnA_Lab3_13.epsillon = 0.1

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 11

4.4.2.3 startPoint list BDAnA_Lab3_13.startPoint = [0.1, 0.1, 0.1]

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 10

4.4.2.4 tEnd int BDAnA_Lab3_13.tEnd = 10

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 7

4.4.2.5 tSize int BDAnA_Lab3_13.tSize = 100000

Див. визначення в файлі BDAnA_Lab3_13.py, рядок 8

5 Файли

5.1 Файл BDAnA_Lab1_13.py

Простори імен

- namespace BDAnA_Lab1_13

Функції

- np.ndarray BDAnA_Lab1_13.calc (X, int t=0)
- None BDAnA_Lab1_13.print_vector_field (ax, x, y, z)
- None BDAnA_Lab1_13.print_trajectories (ax, list starts)

  Будуємо траєкторії.

Змінні

- BDAnA_Lab1_13.x = np.linspace(-1.0, 1.0, 5)
- BDAnA_Lab1_13.y = np.linspace(-1.0, 1.0, 5)
- BDAnA_Lab1_13.z = np.linspace(-1.0, 1.0, 5)
- list BDAnA_Lab1_13.starts
- BDAnA_Lab1_13.fig = plt.figure()
- BDAnA_Lab1_13.ax = fig.gca(projection='3d')

5.2 BDAnA_Lab1_13.py

Див. документацію.

```
00001 #!/usr/bin/env python3
00002
00003 import matplotlib.pyplot as plt
00004 import numpy as np
00005 from scipy.integrate import odeint
00006 from math import sin
00007
00008 x = np.linspace(-1.0, 1.0, 5)
00009 y = np.linspace(-1.0, 1.0, 5)
00010 z = np.linspace(-1.0, 1.0, 5)
00011
00012 starts = [
00013     ([0.8, 0.8, 0.8], 10),
00014     ([0.8, 0.8, -0.8], 10),
00015     ([0.8, -0.8, 0.8], 10),
00016     ([0.8, -0.8, -0.8], 10),
00017     ([-0.8, 0.8, 0.8], 10),
00018     ([-0.8, 0.8, -0.8], 10),
00019     ([-0.8, -0.8, 0.8], 10),
00020     ([-0.8, -0.8, -0.8], 10),
00021 ]
00022
00023
00024 def calc(X, t: int = 0) -> np.ndarray:
00025     x1, x2, x3 = X[0], X[1], X[2]
00026     return np.array([
00027         -x1+2*x2, -2*x2, -2*x1-2*x2-x3
00028     ])
00029
00030
00031 def print_vector_field(ax, x, y, z) -> None:
00032     X, Y, Z = np.meshgrid(x, y, z)
00033     u, v, w = np.zeros(X.shape), np.zeros(Y.shape), np.zeros(Z.shape)
00034     u, v, w = calc([X, Y, Z])
00035     # Друкує на графік векторне поле.
00036     ax.quiver(X, Y, Z, u, v, w, color='r', length=0.1)
00037
00038
```

```
00039
00041 def print_trajectories(ax, starts: list) -> None:
00042     # Обчислюємо траєкторії задані у змінній ‘starts‘
00043     for s, t in starts:
00044         # Задаємо діапазон часу [0, t] та ділить відрізок на 2000 частин.
00045         tspan = np.linspace(0, t, 2000)
00046         # Інтегрує систему до заданого часу ‘t‘ та початкових значень ‘s‘
00047         tr = odeint(calc, s, tspan)
00048         # Друкуємо саму траєкторію
00049         ax.plot3D(tr[:, 0], tr[:, 1], tr[:, 2])
00050         # Друкуємо початок траєкторії у вигляді кружечка
00051         ax.plot3D([tr[0, 0]], [tr[0, 1]], [tr[0, 2]], 'o')
00052         # Друкуємо кінець траєкторії у вигляді квадратика
00053         ax.plot3D([tr[-1, 0]], [tr[-1, 1]], [tr[-1, 2]], 's')
00054
00055
00056 fig = plt.figure()
00057 ax = fig.gca(projection='3d')
00058 print_vector_field(ax, x, y, z)
00059 print_trajectories(ax, starts)
00060 plt.show()
```

5.3 Файл BDAnA_Lab1_13_Analysis.py

Простори імен

- namespace BDAnA_Lab1_13_Analysis

Функції

- np.ndarray BDAnA_Lab1_13_Analysis.calc (X, int t=0)
- None BDAnA_Lab1_13_Analysis.analyze ()

  Аналіз системи.

Змінні

- BDAnA_Lab1_13_Analysis.use_unicode

5.4 BDAnA_Lab1_13_Analysis.py

Див. документацію.

```
00001 #!/usr/bin/env python3
00002
00003 from sympy import *
00004 import numpy as np
00005 init_printing(use_unicode=False)
00006
00007 def calc(X, t: int = 0) -> np.ndarray:
00008     x1, x2, x3 = X[0], X[1], X[2]
00009     return np.array([
00010         -x1+2*x2, -2*x2, -2*x1-2*x2-x3
00011     ])
00012
00013
00014 def analyze() -> None:
00015     t = symbols("t")
00016     x = symbols("x1 x2 x3", cls=Function)
00017     x = [i(t) for i in x]
00018     f = calc(x, t)
00019
00020     # ~ I = Matrix([[i.diff(j) for j in x] for i in f])
00021     pprint(Matrix(f))
00022     I = Matrix(f).jacobian(x)
00023     pprint("Якобіан:")
00024     pprint(I)
00025
00026     solutions = [list(i.values()) for i in solve(f)]
00027     pprint("Особливі точки:")
```

```
00028     pprint(solutions)
00029
00030     for X, Y, Z in solutions:
00031         print()
00032         print("Точка:", X, Y, Z)
00033         pprint(lambdify(x, I, modules="sympy")(X, Y, Z).eigenvals())
00034         eigenvals = list(lambdify(x, I, modules="sympy")(X, Y, Z).eigenvals().keys())
00035         eigenvals = [complex(i) for i in eigenvals]
00036         print("Власні числа:", eigenvals)
00037         if all([i.imag != 0 for i in eigenvals]):
00038             if all([i.real == 0 for i in eigenvals]):
00039                 print("Центр")
00040                 print("Мабуть я у планетарії...")
00041             elif all([i.real < 0 for i in eigenvals]):
00042                 print("Стійкий фокус")
00043                 print("Голова паморочиться... Але рівновагу не втрачаю і голова на місці.")
00044             elif all([i.real > 0 for i in eigenvals]):
00045                 print("Нестійкий фокус")
00046                 print("Голова паморочиться... Зовсім рівновагу втрачаю! Ловіть мене!")
00047             else:
00048                 print("Щось якось обертається, але не знаю як. Розповіси як зустрінимося.")
00049         elif all([i.imag == 0 for i in eigenvals]):
00050             if all([i.real < 0 for i in eigenvals]):
00051                 print("Стійкий вузол")
00052                 print("Очі розлазяться, але десь на одній лінії зустрічаються.")
00053             elif all([i.real > 0 for i in eigenvals]):
00054                 print("Нестійкий вузол")
00055                 print("Очі розлазяться у різні боки.")
00056             else:
00057                 print("Сідло")
00058                 print("Звичайне кінське сідло. Наскільки комфортне не пробував.")
00059         else:
00060             print("А що, так можна було? Комплексні числа у парі зазвичай.")
00061             print("Негайно передзвони та розповіси про такий випадок!")
00062
00063 analyze()
```

5.5 Файл BDAnA_Lab2_13.py

Простори імен

- namespace BDAnA_Lab2_13

Функції

- None BDAnA_Lab2_13.f (float x, float a)

  Відображення з постановки задачі.
- None BDAnA_Lab2_13.draw_diagram ()

  Друкує діаграму.
- None BDAnA_Lab2_13.draw_diagram2 (a, epsillon)

Змінні

- int BDAnA_Lab2_13.firstIters = 1024

  Скільки перших ітерацій пропустити
- int BDAnA_Lab2_13.lastIters = 50

  Скільки останніх ітерацій відобразити на діаграмі
- int BDAnA_Lab2_13.aIters = 2000

  Скільки значень α побудувати для діаграми.
- list BDAnA_Lab2_13.alphaRange = [-2.25, 0.25]

  Діапазон значень α [2.8, 3.8].
- float BDAnA_Lab2_13.xStart = 0.5

  Стартове значення x_0.

5.6 BDAnA_Lab2_13.py

Див. документацію.

```
00001 #!/usr/bin/env python3
00002 import matplotlib.pyplot as plt
00003 import numpy as np
00004
00005
00006 firstIters = 1024
00007
00008 lastIters = 50
00009
00010 aIters = 2000
00011
00012 alphaRange = [-2.25, 0.25]
00013
00016 xStart = 0.5
00017
00018
00019
00023 def f(x: float, a: float) -> None:
00024     return a+x**2+x
00025
00026
00027 def draw_diagram() -> None:
00028     # Знаходимо усі можливі \f$\alpha\f$
00029     As = np.linspace(alphaRange[0], alphaRange[1], aIters)
00030     # Для кожної \f$\alpha\f$ зберегти останні ітерації
00031     grid = np.zeros((lastIters, aIters))
00032
00033     # Робимо ітерації для кожної \f$\alpha\f$
00034     for i in range(aIters):
00035         x = xStart
00036         a = As[i]
00037         # Пропускаємо перші ‘firstIters‘ ітерації.
00038         for _ in range(firstIters):
00039             x = f(x, a)
00040
00041         # Зберінаємо останні ‘lastIters‘ ітерацій.
00042         for j in range(lastIters):
00043             x = f(x, a)
00044             grid[j, i] = x
00045
00046     # Друкуємо графік.
00047     for i in grid:
00048         plt.scatter(As, i, 1, alpha=0.25)
00049     plt.show(block=False)
00050
00051
00052 def draw_diagram2(a, epsillon) -> None:
00053     x = np.zeros(lastIters)
00054     x[0] = xStart
00055
00056     for i in range(1, lastIters):
00057         x[i] = f(x[i-1], a)
00058
00059     xx, yy = np.meshgrid(x, x)
00060     I = (abs(xx - yy) - epsillon) > 0
00061
00062     fig, axs = plt.subplots(1,2)
00063     fig.suptitle(f"x*=ax(1-x), a={a}, e={epsillon}")
00064     axs[0].plot(range(lastIters), x)
00065     axs[1].imshow(I, cmap=plt.cm.gray, origin='lower')
00066     plt.show(block=False)
00067
00068
00069 draw_diagram()
00070 draw_diagram2(-1, 0.01)
00071 draw_diagram2(-1.75, 0.01)
00072 draw_diagram2(-2.25, 0.01)
00073 plt.show()
```

5.7 Файл BDAnA_Lab3_13.py

Простори імен

- namespace BDAnA_Lab3_13

Функції

- def BDAnA_Lab3_13.localmin (x)
- np.ndarray BDAnA_Lab3_13.sys (X, t=0)
- None BDAnA_Lab3_13.draw_attractor ()

Змінні

- int BDAnA_Lab3_13.tEnd = 10
- int BDAnA_Lab3_13.tSize = 100000
- int BDAnA_Lab3_13.dSize = 1000
- list BDAnA_Lab3_13.startPoint = [0.1, 0.1, 0.1]
- float BDAnA_Lab3_13.epsillon = 0.1

5.8 BDAnA_Lab3_13.py

Див. документацію.

```
00001 #!/usr/bin/env python3
00002 import matplotlib.pyplot as plt
00003 import numpy as np
00004 from scipy.integrate import odeint
00005 from nolitsa import data, delay, noise, dimension
00006
00007 tEnd = 10
00008 tSize = 100000
00009 dSize = 1000
00010 startPoint = [0.1, 0.1, 0.1]
00011 epsillon = 0.1
00012
00013
00014 def localmin(x):
00015     return (np.diff(np.sign(np.diff(x))) > 0).nonzero()[0] + 1
00016
00017
00018 def sys(X, t=0) -> np.ndarray:
00019     y1, y2, y3 = X
00020     return np.array([
00021         # ~ 1000-3*x-1000*y**2+10*z**2,
00022         # ~ y+2*z+x*(y+z*4/3),
00023         # ~ -2*y+z+x*(-y*4/3+z)
00024         # ~ 3*x+x*(x-3)*(5*y**2-z**2)/(1+y**2+z**2),
00025         # ~ 2*y-14*z-5*(x-3)*y,
00026         # ~ 14*y+2*z+5*(x-3)*z
00027         # ~ -3*x+140*y**2-z**2,
00028         # ~ y-200*z-140*x*y,
00029         # ~ 200*y+z+x*z
00030         2*y1-20*y3+3*y1**2-2*y2**2-y3**2-2*y2*y3-2*y1*y3,
00031         -0.5*y2+4*y2**2+8*y1*y2+4*y2*y3+4*y1*y3,
00032         20*y1+2*y3+4*y1*y3+2*y2*y3+y3**2
00033     ])
00034
00035
00036 def draw_attractor() -> None:
00037     tspan = np.linspace(0, tEnd, dSize)
00038     ys = odeint(sys, startPoint, tspan, rtol=0.0000000001, atol=0.0000000001)
00039     xx, yy = np.meshgrid(ys[:,0], ys[:,0])
00040     I = (abs(xx - yy) - epsillon) > 0
00041
00042     tspan = np.linspace(0, tEnd, tSize)
00043     yso = ys
00044     ys = odeint(sys, startPoint, tspan)
00045
00046     lag = np.arange(250)
00047     x = ys[:,0]
00048     r = delay.acorr(x, maxtau=250)
00049     i = delay.dmi(x, maxtau=250)
00050
00051     i_delay = localmin(noise.sma(i, hwin=1)) + 1
00052     r_delay = np.argmax(r < 1.0 / np.e)
00053
00054     print(r'Minima of delayed mutual information = %s' % i_delay)
00055     print(r'Autocorrelation time = %d' % r_delay)
00056
00057     dim = np.arange(1, 15 + 1)
```

```
00058     tau_here = (localmin(noise.sma(delay.dmi(x, maxtau=250), hwin=1)) + 1)[0]
00059     tau_here = np.argmax(delay.acorr(yso[:,0], maxtau=250) < 1.0 / np.e)
00060     f = dimension.fnn(yso[:,0], tau=tau_here, dim=dim, window=0, metric='euclidean')
00061
00062     fig = plt.figure(1)
00063     ax = plt.axes(projection='3d')
00064     ax.plot3D(ys[:,0], ys[:,1], ys[:,2], alpha=0.75)
00065     plt.figure(2)
00066     plt.imshow(I, cmap=plt.cm.gray, origin='lower')
00067     plt.figure(3)
00068     plt.subplot(211)
00069     plt.plot(tspan, ys[:,0])
00070     plt.subplot(212)
00071     plt.plot(tspan, ys[:,1])
00072     plt.figure(4)
00073     plt.subplot(121)
00074     plt.plot(ys[:,0], ys[:,1])
00075     plt.subplot(122)
00076     plt.plot(ys[:,0], ys[:,2])
00077     plt.figure(5)
00078     # ~ plt.subplot(121)
00079     # ~ plt.title(r'Time delay = %d' % r_delay)
00080     # ~ plt.xlabel(r'$x(t)$')
00081     # ~ plt.ylabel(r'$x(t + \tau)$')
00082     # ~ plt.plot(ys[:-r_delay,0], ys[r_delay:,0])
00083     # ~ plt.subplot(122)
00084     plt.title(r'Time delay = %d' % i_delay[0])
00085     plt.xlabel(r'$x(t)$')
00086     plt.ylabel(r'$x(t + \tau)$')
00087     plt.plot(ys[:-i_delay[0],0], ys[i_delay[0]:,0])
00088     plt.figure(6)
00089     plt.ylabel(r'Delayed mutual information')
00090     plt.plot(lag, i, i_delay, i[i_delay], 'o')
00091     plt.figure(7)
00092     plt.plot(dim, f[0], dim, f[1], dim, f[2])
00093     plt.xlabel(r'Embedding dimension $d$')
00094     plt.ylabel(r'FNN (%)')
00095     plt.figure(8)
00096     ax = plt.axes(projection='3d')
00097     ax.plot3D(ys[:-2*i_delay[0],0], ys[i_delay[0]:-i_delay[0],0], ys[2*i_delay[0]:,0], alpha=0.75)
00098     print(ys[-1])
00099     plt.show()
00100
00101
00102 draw_attractor()
```

5.9 Файл mainpage.dox

Предметний покажчик