

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

Звіт  
про виконання лабораторної роботи  
з дисципліни “Основи нелінійного аналізу”

Студента групи КМ-71  
Бірук С.В.

Київ – 2020

## ЗМІСТ

1 ПОСТАНОВКА ЗАДАЧІ.....	3
2 АНАЛІТИЧНЕ РОЗВ'ЯЗАННЯ.....	4
3 ПРОГРАМНА РЕАЛІЗАЦІЯ.....	9
ВИСНОВКИ.....	16
Додаток А.....	17

## 1 ПОСТАНОВКА ЗАДАЧІ

Побудувати фазовий портрет для автономної системи, яка має наступний вигляд:

$$\begin{aligned}\frac{dx}{dt} &= x + y + 1 \\ \frac{dy}{dt} &= y + \sqrt{1+2x^2}\end{aligned}$$

## 2 АНАЛІТИЧНЕ РОЗВ'ЯЗАННЯ

$$\begin{aligned} x &= x+y+1 \\ y &= y + \sqrt{1+2x^2} \end{aligned}$$

Особливі точки:  $x = 3 - \sqrt{3}$ ,  $y = \pm \sqrt{2}$

$$\begin{cases} x+y+1=0 \\ y+\sqrt{1+2x^2}=0 \end{cases} \Rightarrow \begin{cases} y = -(x+1) \\ 1+2x^2 = (x+1)^2 \end{cases}$$

$$1+2x^2 = x^2 + 2x + 1$$

$$x^2 - 2x = 0$$

$$\begin{cases} x_1 = 0 \\ y_1 = -1 \end{cases} \quad \begin{cases} x_2 = 2 \\ y_2 = -3 \end{cases}$$

Точка:  $(0; -1)$

$$\begin{cases} x = \varepsilon_1 \\ y = \varepsilon_2 - 1 \end{cases} \Rightarrow \begin{cases} \varepsilon_1 = \varepsilon_1 + \varepsilon_2 \\ \varepsilon_2 = \varepsilon_2 - 1 + \sqrt{1+2\varepsilon_1^2} \end{cases}$$

#  $\sqrt{1+2\varepsilon_1^2} \approx 1 + \frac{0.5}{1} \cdot 2\varepsilon_1^2$

$$\begin{cases} \varepsilon_1 = \varepsilon_1 + \varepsilon_2 \\ \varepsilon_2 = \varepsilon_2 \end{cases}$$

Рис. 2.1 – Визначення особливих точок

$$\begin{pmatrix} 1-\lambda & 1 \\ 0 & 1-\lambda \end{pmatrix} \Rightarrow (1-\lambda)^2 = 0$$

$\lambda_1, \lambda_2 = 1, \alpha = 2 \Rightarrow$  Вироджений лінійний

$$\lambda = 1, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \Rightarrow V = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\# \quad \varepsilon_2 d\varepsilon_1 - (\varepsilon_1 + \varepsilon_2) d\varepsilon_2 = 0, \quad \varepsilon_2' = \frac{\varepsilon_2}{\varepsilon_1 + \varepsilon_2}$$

Карл.  $\varepsilon_1$ : Карл.  $\varepsilon_2$ :

$$\varepsilon_2 = 0$$

$$\varepsilon_2 = \frac{k\varepsilon_1}{1-k}$$

$$\begin{array}{c|cc|c} k & -1 & 2 \\ \hline \varepsilon_2 & -\frac{1}{2}\varepsilon_1 & -2\varepsilon_1 \\ \hline & -\frac{\pi}{6} & \approx 60^\circ \end{array}$$

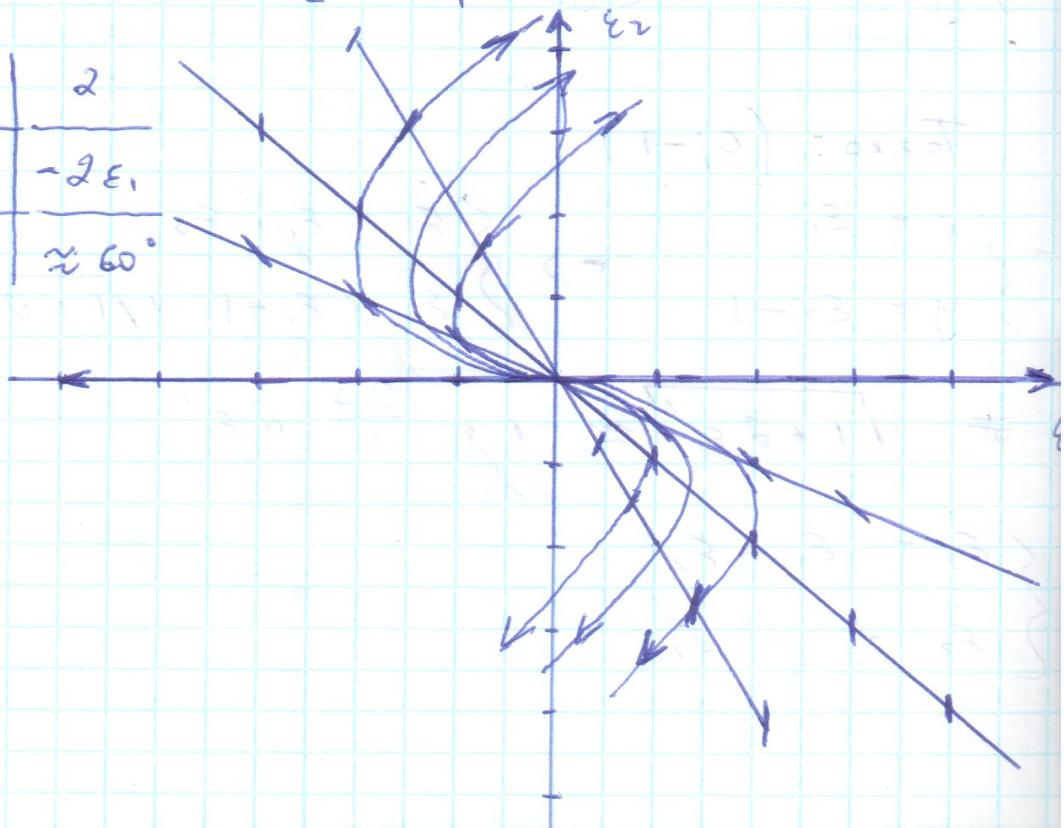


Рис. 2.2 – Дослідження точки  $(0; -1)$

Точка!  $(2, -3)$

$$\begin{cases} x = 2 + \varepsilon_1 \\ y = -3 + \varepsilon_2 \end{cases}$$

$$\begin{cases} \dot{\varepsilon}_1 = 2 + \varepsilon_1 - 3 + \varepsilon_2 + 1 \\ \dot{\varepsilon}_2 = \varepsilon_2 - 3 + \sqrt{1 + 2(2 + \varepsilon_1)^2} \end{cases} \Rightarrow \begin{cases} \dot{\varepsilon}_1 = \varepsilon_1 + \varepsilon_2 + 3 \\ \dot{\varepsilon}_2 = \varepsilon_2 - 3 + \sqrt{9 + 8\varepsilon_1 + 2\varepsilon_1^2} \end{cases}$$

$$\# \sqrt{9 + 8\varepsilon_1 + 2\varepsilon_1^2} = 3(1 + \frac{4}{9}\varepsilon_1 + \frac{2}{9}\varepsilon_1^2)^{0.5} \approx$$

$$\approx \varepsilon_2 - 3 + 3(1 + \frac{4}{9}\varepsilon_1) = \varepsilon_2 + \frac{4}{3}\varepsilon_1$$

$$\begin{cases} \dot{\varepsilon}_1 = \varepsilon_1 + \varepsilon_2 \\ \dot{\varepsilon}_2 = \frac{4}{3}\varepsilon_1 + \varepsilon_2 \end{cases}$$

$$\begin{pmatrix} 1 & 1 \\ \frac{4}{3} & 1 \end{pmatrix} \Rightarrow (1 - \lambda)^2 - \frac{4}{3} = 3\lambda^2 - 6\lambda - 1 = 0$$

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{6 \pm \sqrt{37}}{6} = 1 \pm \frac{2}{3}\sqrt{3}$$

Оскільки  $\lambda_1 > 0, \lambda_2 < 0$ , отже відно

$$\lambda_1 = 1 + \frac{2}{3}\sqrt{3} > 0$$

$$\begin{pmatrix} -\frac{2}{3}\sqrt{3} & 1 \\ \frac{4}{3} & -\frac{2}{3}\sqrt{3} \end{pmatrix} \Rightarrow V_1 = \begin{pmatrix} +\frac{\sqrt{3}}{2} \\ 1 \end{pmatrix}$$

$$\lambda_2 = 1 - \frac{2}{3}\sqrt{3} < 0$$

$$\begin{pmatrix} \frac{2}{3}\sqrt{3} & 1 \\ \frac{4}{3} & -\frac{2}{3}\sqrt{3} \end{pmatrix} \Rightarrow V_2 = \begin{pmatrix} -\frac{\sqrt{3}}{2} \\ 1 \end{pmatrix}$$

Рис. 2.3 – Дослідження точки  $(2; -3)$

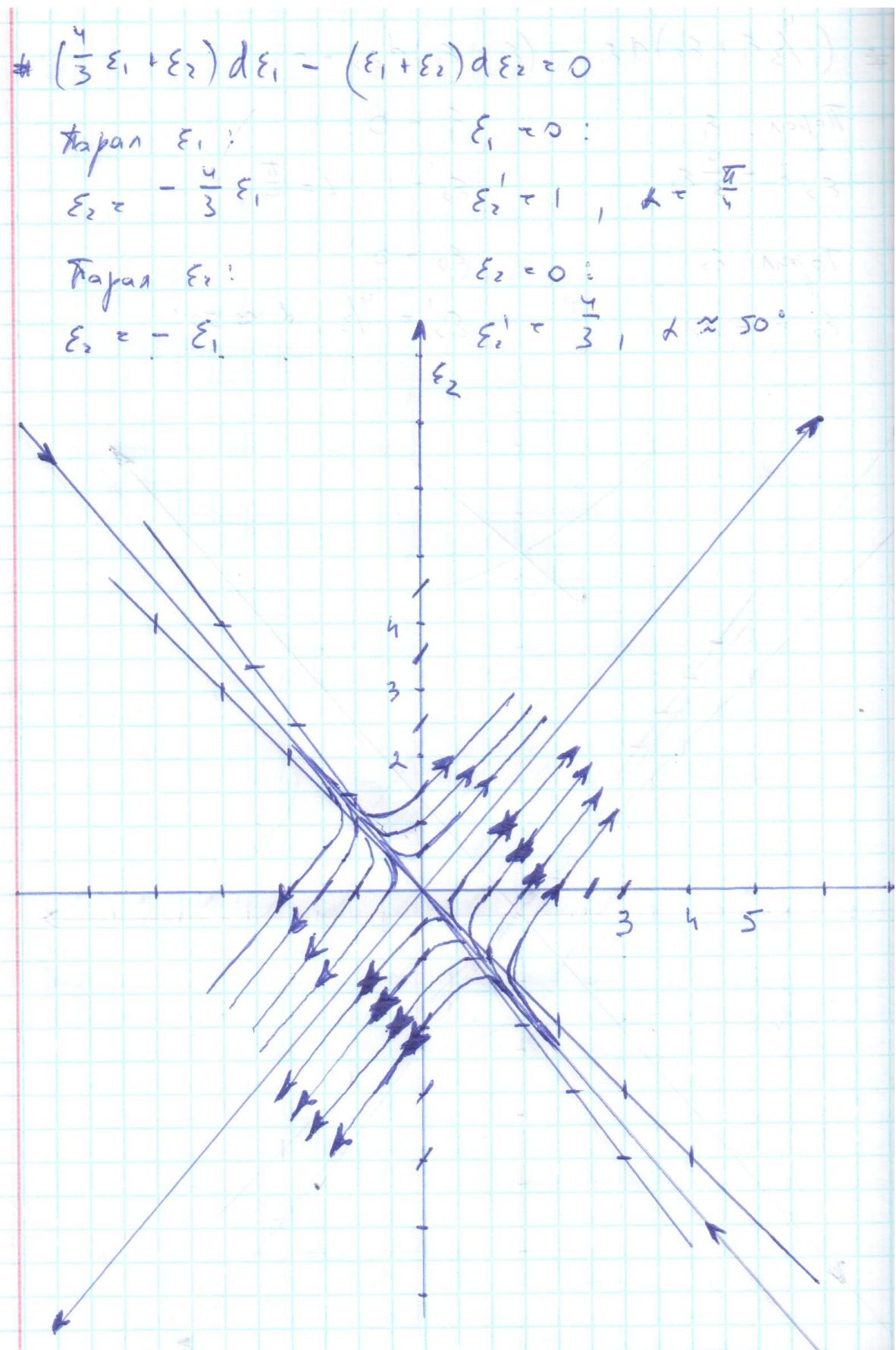


Рис. 2.4 Фазовий портрет в околі точки  $(2; -3)$

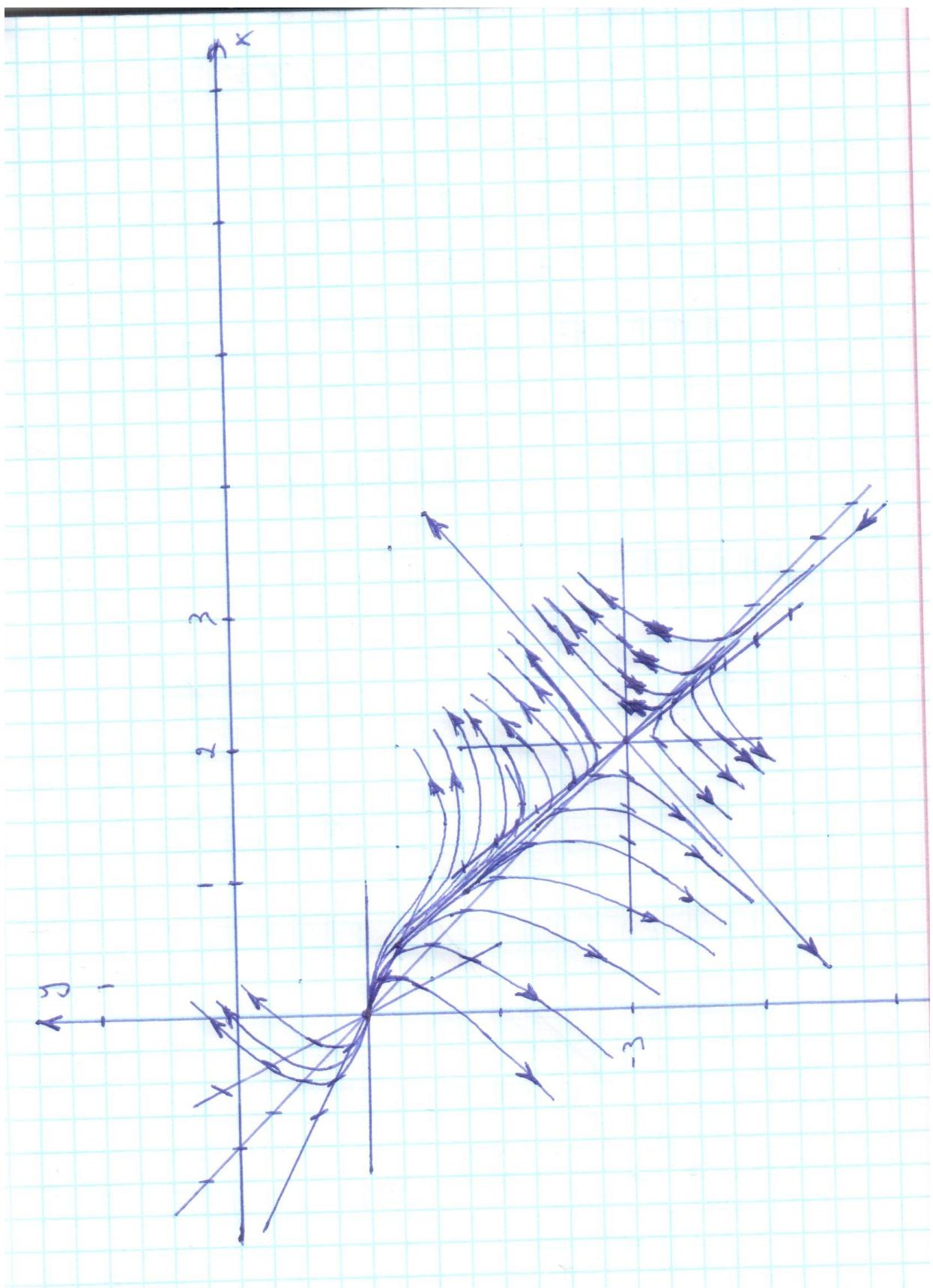


Рис. 3.5 Фазовий портрет нелінійної системи

## З ПРОГРАМНА РЕАЛІЗАЦІЯ

Пошук фазових портретів за допомогою програмного забезпечення передбачає виконання наступної послідовності дій:

- Визначення особливих точок
- Лінеразація системи в околі особливої точки
- Пошук власних чисел та власних векторів
- Побудова фазового портрету

Для подальших обрахунків будемо використовувати наступні бібліотеки:

- numpy — для векторизації обчислень
- sympy — для пошуку особливих точок, а також обчислення похідних
- matplotlib — для візуалізації результатів

Пошук особливих точок програмним шляхом зображенено на рис. 3.1

### The equilibria

```
In [2]: x, y = sympy.symbols('x, y')
In [3]: eq1 = x + y + 1
         eq2 = y + sympy.sqrt(1 + 2 * x**2)
In [4]: equilibria = sympy.solve([eq1, eq2], [x, y])
equilibria
Out[4]: [(0, -1), (2, -3)]
```

Рис. 3.1 Пошук особливих точок

Для визначення типу фазових портретів в околі кожної із особливих точок, необхідно провести заміну змінних та провести лінеразацію. Даний пункт не обов'язковий для побудови фазових портретів програмним шляхом, однак в цілях

дослідження важливо знати власні числа та власні вектори для визначення типу характеру фазових кривих в околі точки. На рис. 3.2 зображенено процес заміни змінних.

### Make variable replacement

```
In [5]: e1, e2 = sympy.symbols('e1, e2')

In [6]: eq1_point1 = eq1.subs(x, e1 + equilibria[0][0]).subs(y, e2 + equilibria
eq2_point1 = eq2.subs(x, e1 + equilibria[0][0]).subs(y, e2 + equilibria
          ◀          ▶

In [7]: eq1_point1, eq2_point1

Out[7]: (e1 + e2, e2 + sqrt(2*e1**2 + 1) - 1)

In [8]: eq1_point2 = eq1.subs(x, e1 + equilibria[1][0]).subs(y, e2 + equilibria
eq2_point2 = eq2.subs(x, e1 + equilibria[1][0]).subs(y, e2 + equilibria
          ◀          ▶

In [9]: eq1_point2, eq2_point2

Out[9]: (e1 + e2, e2 + sqrt(2*(e1 + 2)**2 + 1) - 3)
```

Рис. 3.2 Проведення заміни змінних

Наступна частина програмної реалізації реалізує лінерезацію за допомогою розкладення в ряд Тейлора. Відразу після цього етапу відбувається пошук власних чисел, їх кратності та власних векторів.

**Point №1**

```
In [10]: leq1_point1 = eq1_point1.subs(e1, 0).subs(e2, 0) + eq1_point1.diff(e1).
leq2_point1 = eq2_point1.subs(e1, 0).subs(e2, 0) + eq2_point1.diff(e1).
In [11]: leq1_point1, leq2_point1
Out[11]: (e1 + e2, e2)

In [12]: A1, _ = sympy.linear_eq_to_matrix([sympy.expand(sympy.Eq(leq1_point1, 0)
sympy.expand(sympy.Eq(leq2_point1, 0))
In [13]: A1
Out[13]: [1 1]
[0 1]

In [14]: A1.eigenvals()
Out[14]: [(1,
2,
[Matrix([
[1],
[0]]))]]
```

Рис. 3.3 Проведення лінерезації та пошук власних чисел та векторів для точки (0; -1)

**Point №2**

```
In [15]: leq1_point2 = eq1_point2.subs(e1, 0).subs(e2, 0) + eq1_point2.diff(e1).
leq2_point2 = eq2_point2.subs(e1, 0).subs(e2, 0) + eq2_point2.diff(e1).
In [16]: leq1_point2, leq2_point2
Out[16]: (e1 + e2, 4*e1/3 + e2)

In [17]: A2, _ = sympy.linear_eq_to_matrix([sympy.expand(sympy.Eq(leq1_point2, 0)
sympy.expand(sympy.Eq(leq2_point2, 0))
In [18]: A2
Out[18]: [1 1]
[4/3 1]

In [40]: A2.eigenvals()
Out[40]: [(1 - 2*sqrt(3)/3,
1,
[Matrix([
[-sqrt(3)/2,
[1]])]),
(1 + 2*sqrt(3)/3,
1,
[Matrix([
[sqrt(3)/2,
[1]])]])]
```

Рис. 3.4 Проведення лінерезації та пошук власних чисел та векторів для точки (2; -3)

Для досягнення цілі побудови фазового портрету не потрібно проводити етапи, які описані вище. Але для дослідницьких цілей це досить корисна інформація. Частина програмної реалізації, яка здійснює відображення фазових кривих зображено на рис. 3.5

```
In [20]: def f(xy, t=0):
    x, y = xy
    return [x + y + 1,
            y + np.sqrt(1 + 2 * x**2)]

In [27]: x_min, x_max = -4, 7
y_min, y_max = -7, 4

xx = np.linspace(x_min, x_max, 25)
yy = np.linspace(y_min, y_max, 25)

X, Y = np.meshgrid(xx, yy)
DX, DY = f([X, Y])

M = (np.hypot(DX, DY))
M[M == 0] = 1.

NDX = DX / M
NDY = DY / M

fig, ax = plt.subplots(1, 1, figsize=(12, 12))
ax.quiver(X, Y, NDX, NDY, M, pivot='mid')

color = 2 * np.log(np.hypot(DX, DY))
ax.streamplot(X, Y, DX, DY, color=color, linewidth=2, cmap=plt.cm.inferno,
              density=2, arrowstyle='-', arrowsize=1.5)

ax.add_artist(Circle((0, -1), 0.15, color="#aa0000"))
ax.add_artist(Circle((2, -3), 0.15, color="#aa0000"))

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('Фазові криві')
```

Рис. 3.5 Відображення фазових кривих

Дана програмна реалізація не здійснює пошук розв'язку нелінійної системи диференційних рівнянь, а на деякому розбитті області здійснює пошук вектора зміщення відносно кожної із точок розбиття. Результати зображені на рис. 3.6

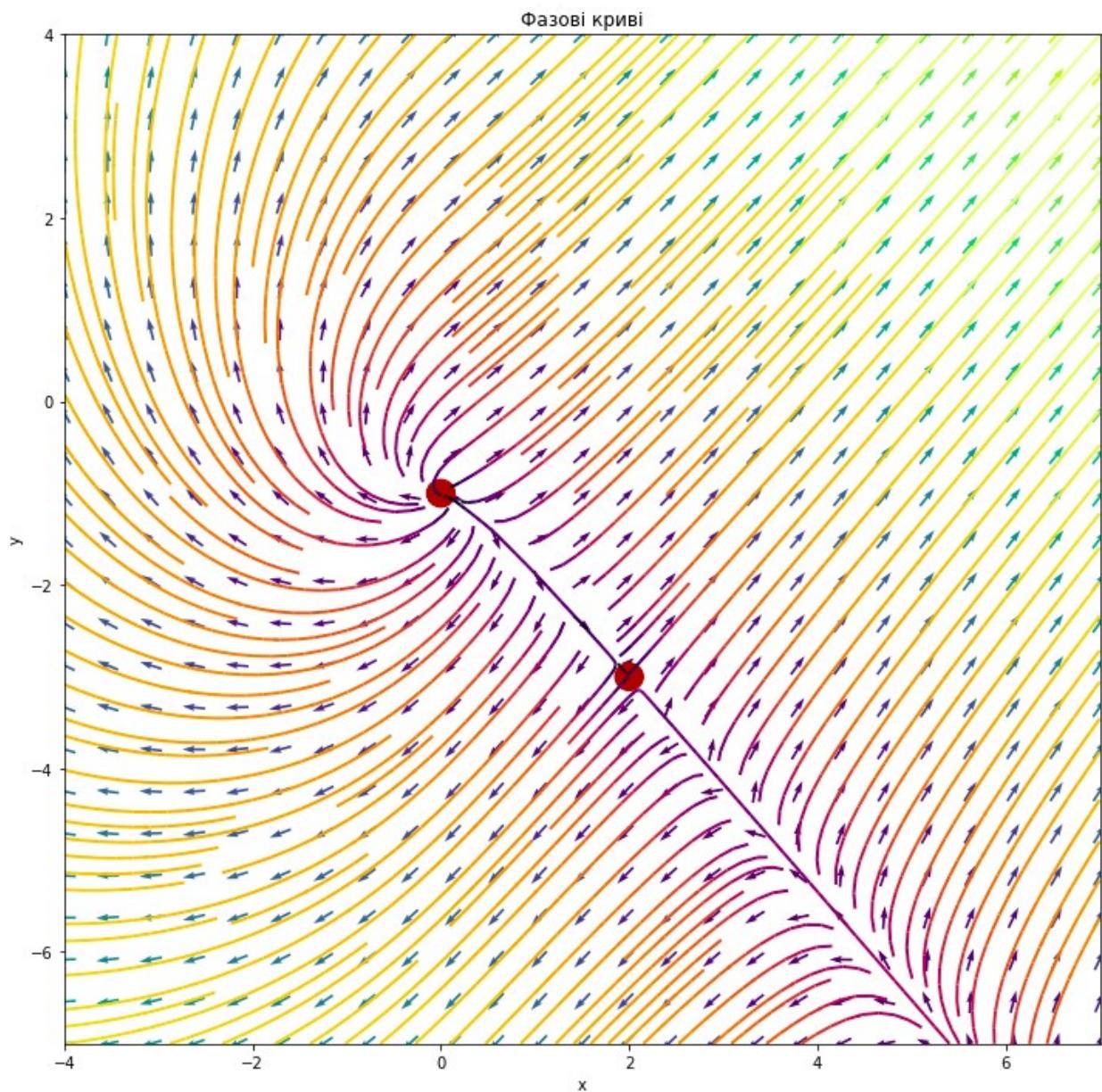


Рис. 3.6 Фазовий портрет

В дослідницьких цілях досить корисно визначити характер поведінки фазових кривих в околі кожної особливої точки. Дані представлення зображенено на рис. 3.7 та 3.8.

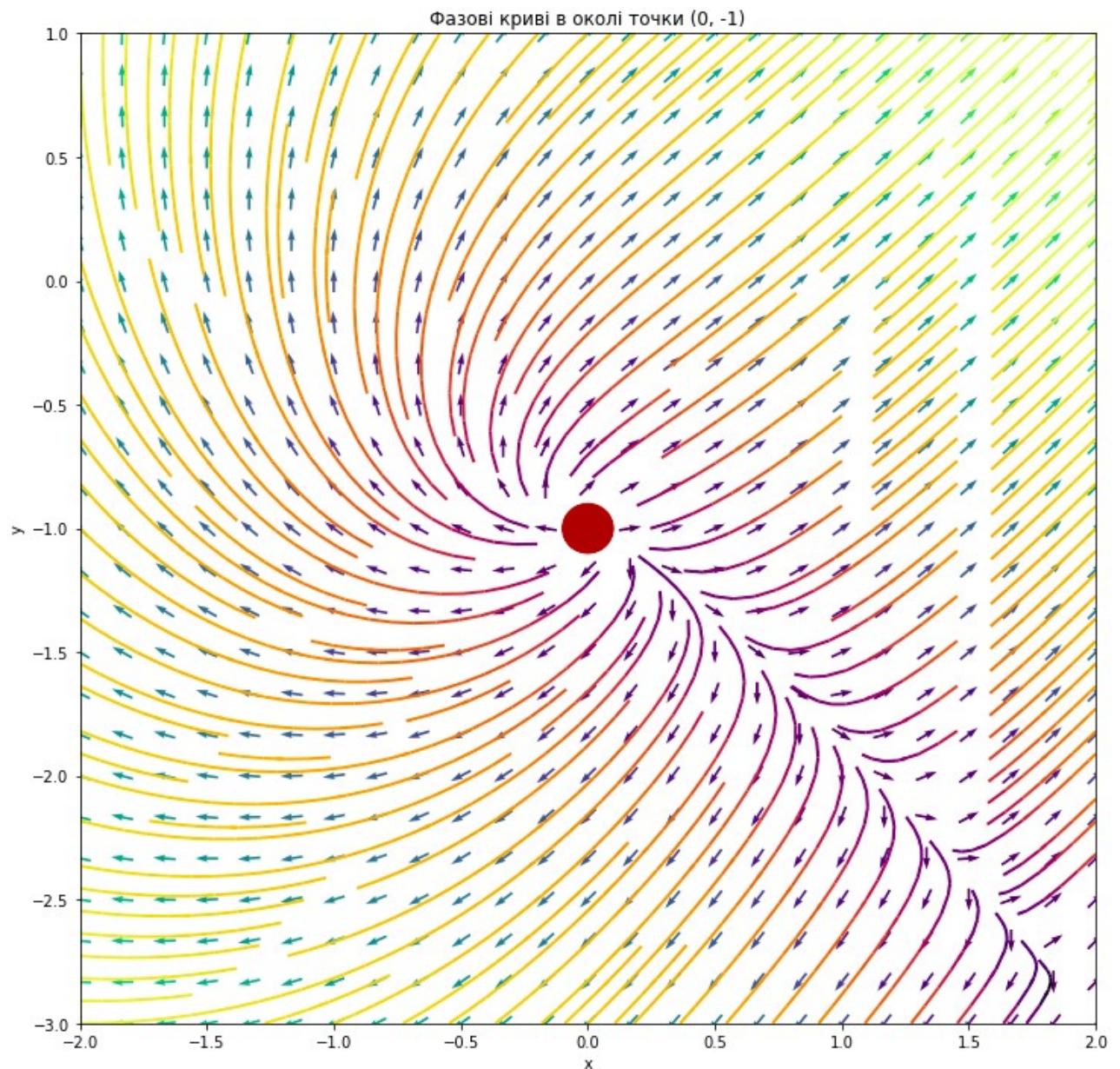


Рис. 3.7 Фазовий портрет в околі особливої точки  $(0; -1)$

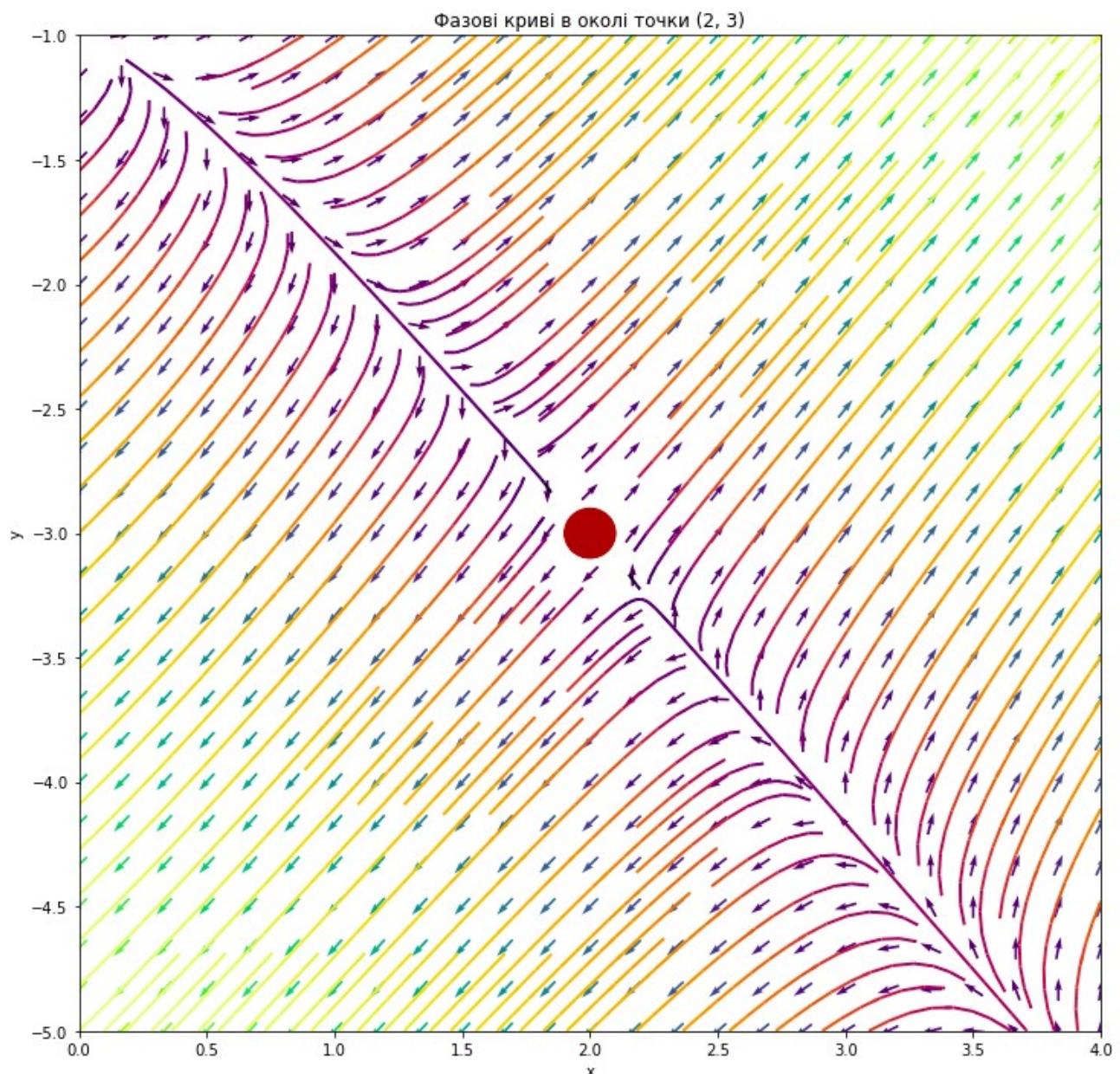


Рис. 3.8 Фазовий портрет в околі особливої точки  $(2; -3)$

## ВИСНОВКИ

В межах даної лабораторної роботи було розроблено програмне забезпечення, яке дозволяє швидко визначити характер поведінки фазових кривих нелінійної системи диференційних рівнянь.

При виконанні було отримано досить не погане приближення фазових кривих. Однак, в околах особливих точок та вздовж кожної із бісеректрис існують зони де не вдалося обчислити вектор зміщення. Дану проблему можна вирішити за допомогою знаходження чисельними методами розв'язку системи задавши початкові координати в околі особливих точок або бісеректриси. Однак виникає ряд інших проблем.

Для визначення характеру поведінки графічне представлення фазових портретів є досить інформативним. Але для визначення поведінки конкретної фазової кривої із заданими початковими координатами краще використовувати чисельні методи або ж якщо це можливо символне програмування.

## Додаток А

```

import sympy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle

def f(xy, t=0):
    x, y = xy
    return [x + y + 1,
            y + np.sqrt(1 + 2 * x**2)]

x_min, x_max = -4, 7
y_min, y_max = -7, 4

xx = np.linspace(x_min, x_max, 25)
yy = np.linspace(y_min, y_max, 25)

X, Y = np.meshgrid(xx, yy)
DX, DY = f([X, Y])

M = (np.hypot(DX, DY))
M[M == 0] = 1.

NDX = DX / M
NDY = DY / M

fig, ax = plt.subplots(1, 1, figsize=(12, 12))
ax.quiver(X, Y, NDX, NDY, M, pivot='mid')

color = 2 * np.log(np.hypot(DX, DY))
ax.streamplot(X, Y, DX, DY, color=color, linewidth=2, cmap=plt.cm.inferno,
              density=2, arrowstyle='-', arrowsize=1.5)

ax.add_artist(Circle((0, -1), 0.15, color='#aa0000'))
ax.add_artist(Circle((2, -3), 0.15, color='#aa0000'))

ax.set_xlabel('x')

```

```

ax.set_ylabel('y')
ax.set_title('Фазові криві')

x_min, x_max = -2, 2
y_min, y_max = -3, 1

xx = np.linspace(x_min, x_max, 25)
yy = np.linspace(y_min, y_max, 25)

X, Y = np.meshgrid(xx, yy)
DX, DY = f([X, Y])

M = (np.hypot(DX, DY))
M[ M == 0] = 1.

NDX = DX / M
NDY = DY / M

fig, ax = plt.subplots(1,1, figsize=(12,12))
ax.quiver(X, Y, NDX, NDY, M, pivot='mid')

color = 2 * np.log(np.hypot(DX, DY))
ax.streamplot(X, Y, DX, DY, color=color, linewidth=2, cmap=plt.cm.inferno,
              density=1.5, arrowstyle='-', arrowsize=1.5)

ax.add_artist(Circle((0, -1), 0.1, color="#aa0000"))

ax.set_xlabel('x')
ax.set_ylabel("y")
ax.set_title("Фазові криві в околі точки (0, -1)")

x_min, x_max = 0, 4
y_min, y_max = -5, -1

xx = np.linspace(x_min, x_max, 25)
yy = np.linspace(y_min, y_max, 25)

X, Y = np.meshgrid(xx, yy)
DX, DY = f([X, Y])

```

```
M = (np.hypot(DX, DY))
M[ M == 0] = 1.

NDX = DX / M
NDY = DY / M

fig, ax = plt.subplots(1,1, figsize=(12,12))
ax.quiver(X, Y, NDX, NDY, M, pivot='mid')

color = 2 * np.log(np.hypot(DX, DY))
ax.streamplot(X, Y, DX, DY, color=color, linewidth=2, cmap=plt.cm.inferno,
              density=1.5, arrowstyle='-' , arrowsize=1.5)

ax.add_artist(Circle((2, -3), 0.1, color="#aa0000"))

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title("Фазові криві в околі точки (2, -3)!")
```