

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №1
з дисципліни «Теорія керування»
на тему
«Навігаційна задача швидкодії»

Виконав:

студент групи КМ-01
Романецький М.С.

Перевірили:

Професор ПМА ФПМ
Норкін В.І.
Асистент ПМА ФПМ
Жук І.С.

Зміст

Постановка задачі.....	3
Теоретична частина.....	4
Основна частина.....	5
Дослідження параметру s_0 :	6
Дослідження параметру v :	7
Дослідження параметру l :	8
Дослідження параметру φ :	9
Дослідження параметру N :	10
Висновок.....	13
Додаток А – Код програми	14

Постановка задачі

1. Задати вихідні дані $s_0 = \sqrt{n}$, $v = \sqrt{n}$, $l = n$, $\varphi = n\pi/25$, $f(x_2) = x_2$, де n - номер студента у списку групи.
2. Запрограмувати рекурентні співвідношення в системі Python.
3. Побудувати графік траєкторії судна
(а також цілі для випадку завдань В, Г). На графіках вказувати:
А. назву графіка і назви осей координат,
В. точку призначення і значення фіксованих параметрів,
С. легенду (який колір відповідає якомусь значенню параметра),
D. час досягнення цілі.
4. Дослідити залежність траєкторії от варіації параметрів s_0 , v , l , φ , N
5. При яких значеннях параметрів, в тому числі за скільки ітерацій і за який час, корабель досягає точки призначення (для даної (міопичної) стратегії управління)?
6. Підготувати звіт про роботу в електронному вигляді (з графіками, висновками і лістингом програми).
7. Надіслати звіт викладачеві на електронну адресу.

Варіант роботи відповідає порядковому номеру студента – 16 варіант

Отже, у ході роботи $n = 16$.

Теоретична частина

Шукатимемо (міопичне = короткозоре) управління $(u_1(t), u_2(t))$ як рішення наступного завдання:

$$\begin{aligned} & \left(x_1^* - x_1(t + \tau) \right)^2 + \left(x_2^* - x_2(t + \tau) \right)^2 = \\ & = \left(x_1^* - \left(x_1(t) + (s(x_2(t)) + v \cdot u_1) \tau \right) \right)^2 + \left(x_2^* - (x_2(t) + v u_2 \tau) \right)^2 \rightarrow \min_{u: u_1^2 + u_2^2 = 1}. \end{aligned}$$

Рішення даної задачі оптимізації відбувається методом множників Лагранжа, тому розглянемо задачу:

$$\left(x_1^* - \left(x_1(t) + (s(x_2(t)) + v \cdot u_1) \tau \right) \right)^2 + \left(x_2^* - (x_2(t) + v u_2 \tau) \right)^2 + \lambda (u_1^2 + u_2^2 - 1) \rightarrow \min_{u_1, u_2}.$$

Необхідні умови екстремуму мають наступний вигляд:

$$\begin{aligned} & -2 \left(x_1^* - \left(x_1(t) + (s(x_2(t)) + v \cdot u_1) \tau \right) \right) v \tau + 2 \lambda u_1 = 0, \\ & -2 \left(x_2^* - (x_2(t) + v u_2 \tau) \right) v \tau + 2 \lambda u_2 = 0, \\ & u_1^2 + u_2^2 = 1. \end{aligned}$$

Звідки отримуємо:

$$\begin{aligned} u_1(t) &= \frac{\left(x_1^* - x_1(t) - s(x_2(t)) \cdot \tau \right) \cdot v \cdot \tau}{\lambda(t) + v^2 \tau^2}, \quad u_2(t) = \frac{\left(x_2^* - x_2(t) \right) v \tau}{\lambda(t) + v^2 \tau^2}, \\ \lambda(t) &= \left(\left(x_1^* - x_1(t) - s(x_2(t)) \tau \right)^2 + \left(x_2^* - x_2(t) \right)^2 \right)^{1/2} v \tau - v^2 \tau^2; \\ x_1^* &= l \cdot \cos \varphi, \quad x_2^* = l \cdot \sin \varphi. \end{aligned}$$

Основна частина

Визначимо вхідні дані відповідно до варіанту:

$$s_0 = \sqrt{16}, v = \sqrt{16}, l = 16, \varphi = \frac{16 * \pi}{25}, f(x_2) = x_2$$

Але при таких вхідних даних човен не зміг потрапити у кінцеву точку.

Тому було збільшено швидкість корабля $v = 100$.

За такої швидкості човен зміг дібратись до заданої точки.

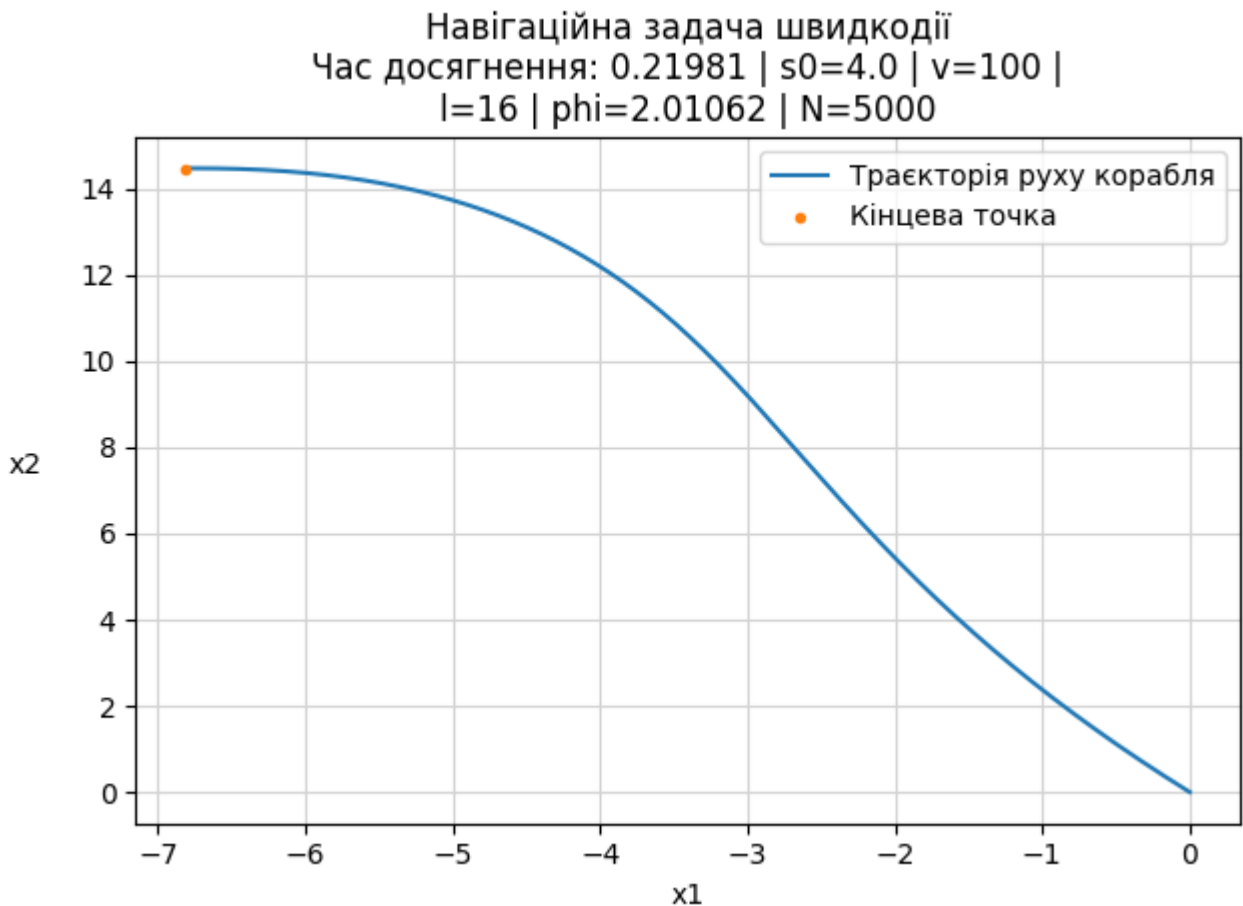


Рис. 1 – Траєкторія руху корабля

Далі були проведені дослідження залежності траєкторій корабля від зміни різних параметрів.

Дослідження параметру s_0 :

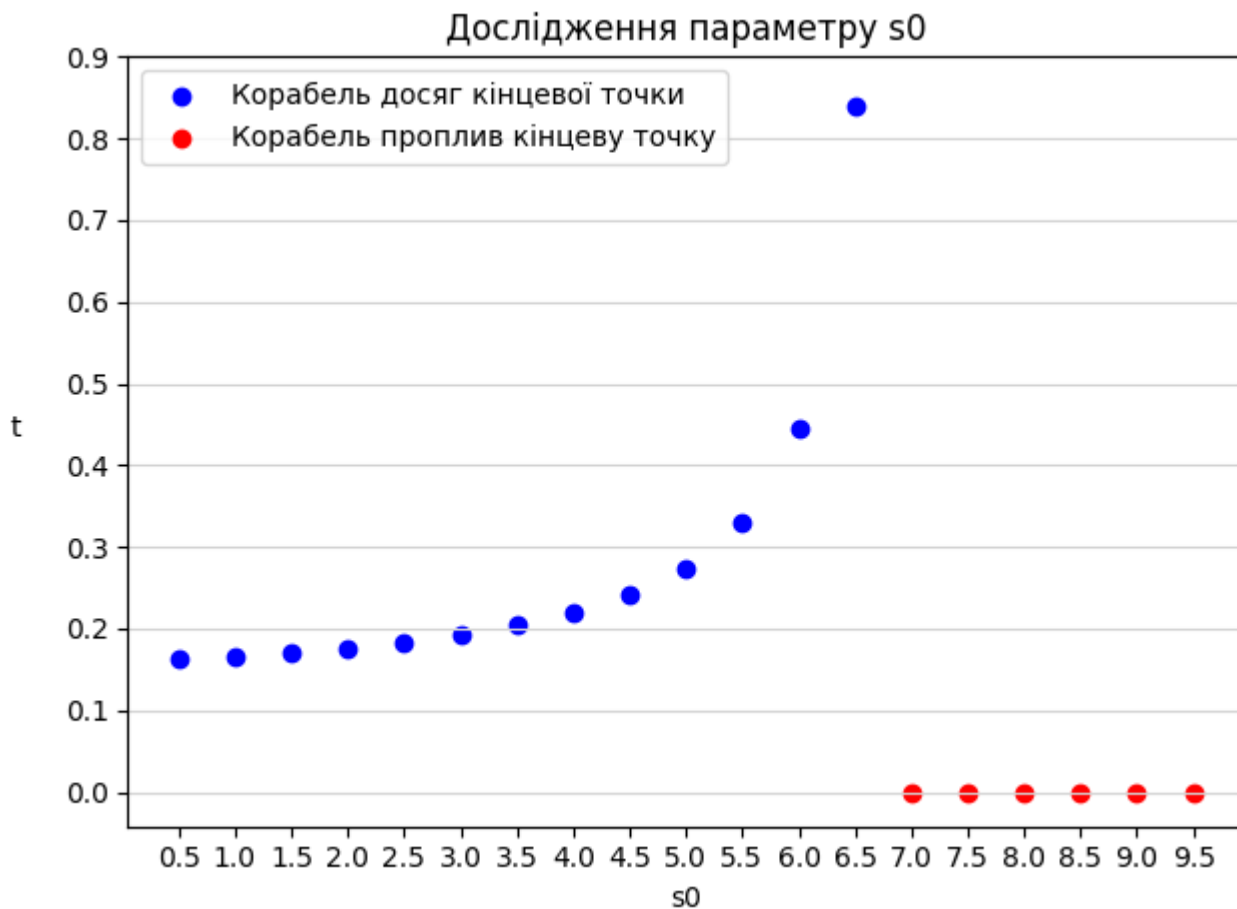


Рис. 2 – Залежність часу досягнення кінцевої точки від швидкості течії s_0

Тут використовувались такі значення інших параметрів:

$$v = 100, l = 16, \varphi = \frac{16\pi}{25}, f(x_2) = x_2, N = 5000$$

$$s_0 = [0.5, 10] \text{ з кроком } 0.5.$$

Не дивлячись на досить велику швидкість, човен здатний досягти кінцевої точки не при всіх значеннях течії, інакше його просто зносить течією.

Дослідження параметру v :

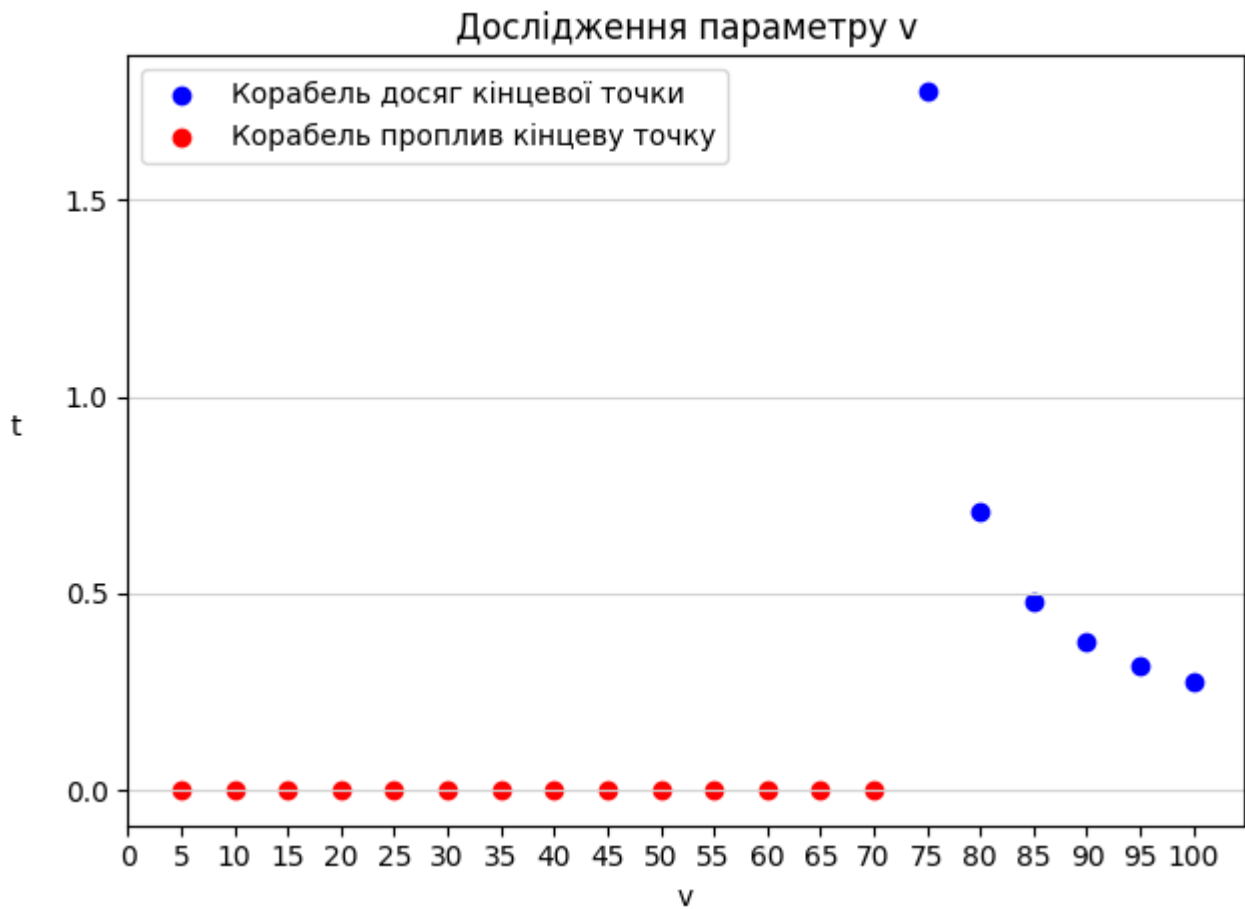


Рис. 3 – Залежність часу досягнення кінцевої точки від швидкості корабля v

Тут використовувались такі значення інших параметрів:

$$s_0 = 5, l = 16, \varphi = \frac{16\pi}{25}, f(x_2) = x_2, N = 5000$$

$v = [5, 100]$ з кроком 5.

Корабель здатний досягти кінцевої точки лише за швидкості в 75 і більше. В іншому випадку він не може подолати швидкість течії.

Дослідження параметру l :



Рис. 4 – Залежність часу досягнення кінцевої точки від відстані до точки 1

Тут використовувались такі значення інших параметрів:

$$s_0 = 5, v = 100, \varphi = \frac{16\pi}{25}, f(x_2) = x_2, N = 5000$$

$l = [1, 31]$ з кроком 1.

В даному випадку можна звернути увагу, що кінцева точка повинна бути не надто далеко від початкової, оскільки функція залежності швидкості течії від координати x_2 – лінійна, а отже чим далі буде знаходитися точка, тим більшою буде швидкість течії.

Дослідження параметру φ :

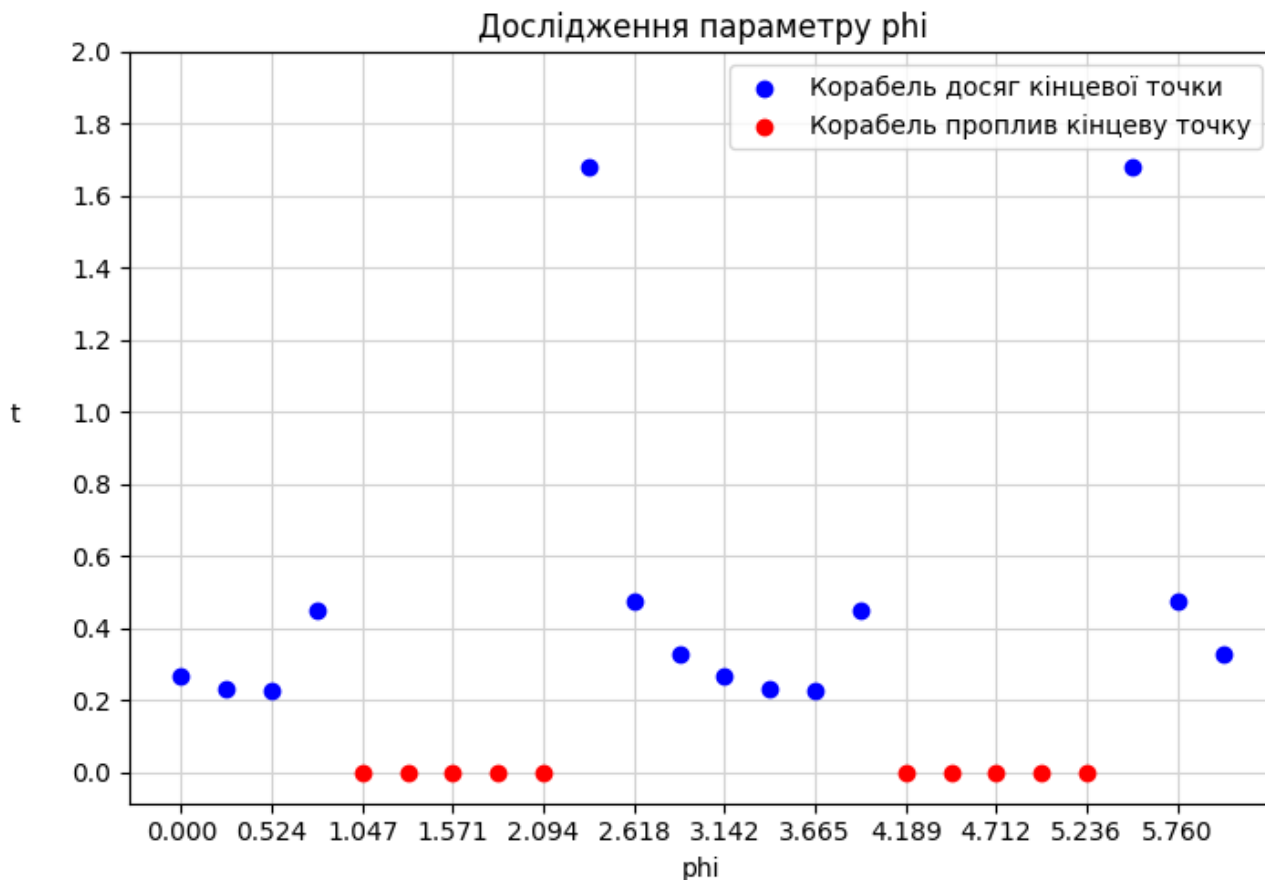


Рис. 5 – Залежність часу досягнення кінцевої точки від кута φ

Тут використовувались такі значення інших параметрів:

$$s_0 = 5, v = 60, l = 16, f(x_2) = x_2, N = 5000$$

$$\varphi = [0, 2\pi] \text{ з кроком } \frac{\pi}{12}.$$

Проаналізувавши даний графік, можна зробити висновок, що корабель може досягнути кінцевої точки за даних параметрів, якщо кут φ знаходиться приблизно в межах $\pm \frac{\pi}{4}$ та $\pi \pm \frac{\pi}{4}$. Дані результати можна пояснити видом залежності швидкості течії від координати x_2 . Оскільки чим більше росте x_2 , тим швидше збільшується швидкість течії, то очевидно, що при кутах, близьких до $\frac{\pi}{2}$ або $\frac{3\pi}{2}$, корабель не зможе досягти кінцевої точки за наявної швидкості течії.

Дослідження параметру N:

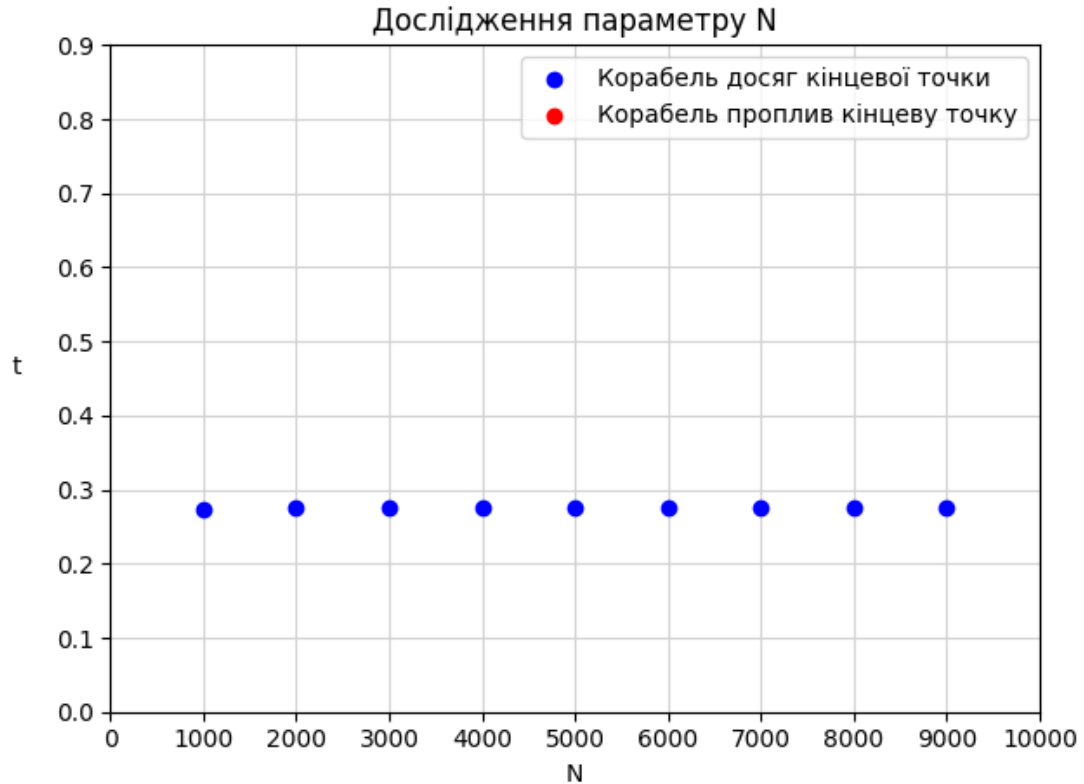


Рис. 6 – Залежність часу досягнення кінцевої точки від параметра N

Тут використовувались такі значення інших параметрів:

$$s_0 = 5, v = 100, l = 16, \varphi = \frac{16\pi}{25}, f(x_2) = x_2$$

$N = [1000, 10000]$ з кроком 1000.

З графіку видно, що корабель допливає до кінцевої точки з приблизно однаковим часовим не зважаючи на зміну даного параметру. Було протестовано інші набори вхідних даних, але результат завжди залишався одним – корабель або допливав до кінцевої точки для всіх значень N або не допливав. Було також випробувано більш широкий діапазон значень параметру N, однак результат це також не змінило. Тому можна зробити висновок, що величина даного параметру не є важливою для побудови траєкторії руху корабля.

Для більш наочного представлення було побудовано траєкторії руху корабля до кінцевої точки для різних наборів вихідних даних:

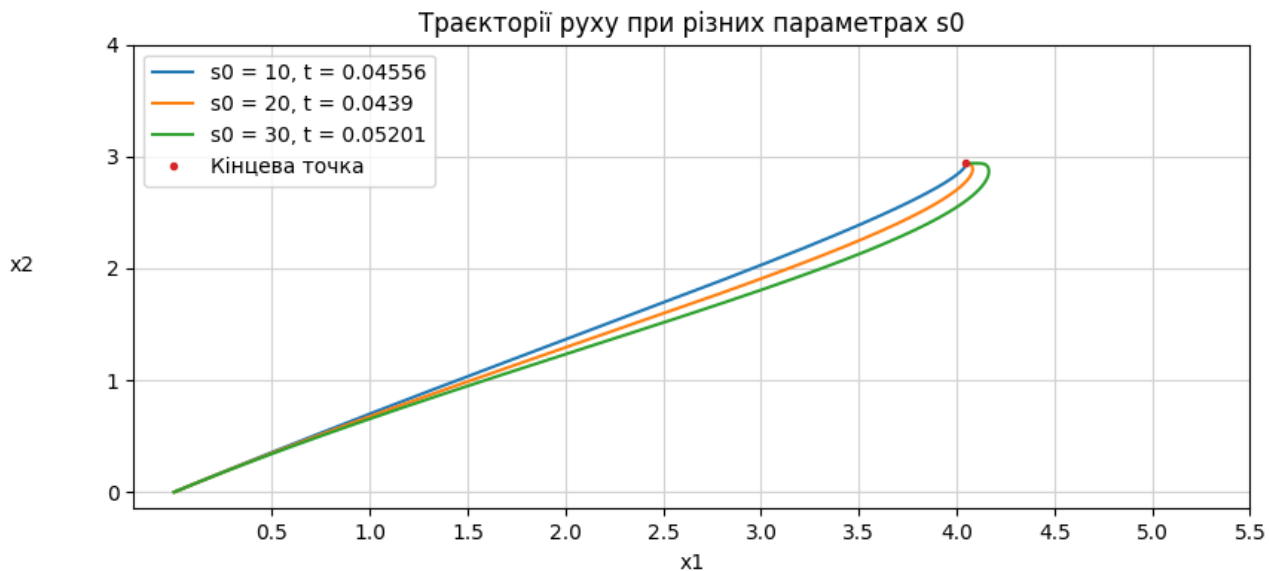


Рис. 7 – траєкторії руху корабля при різних параметрах s_0

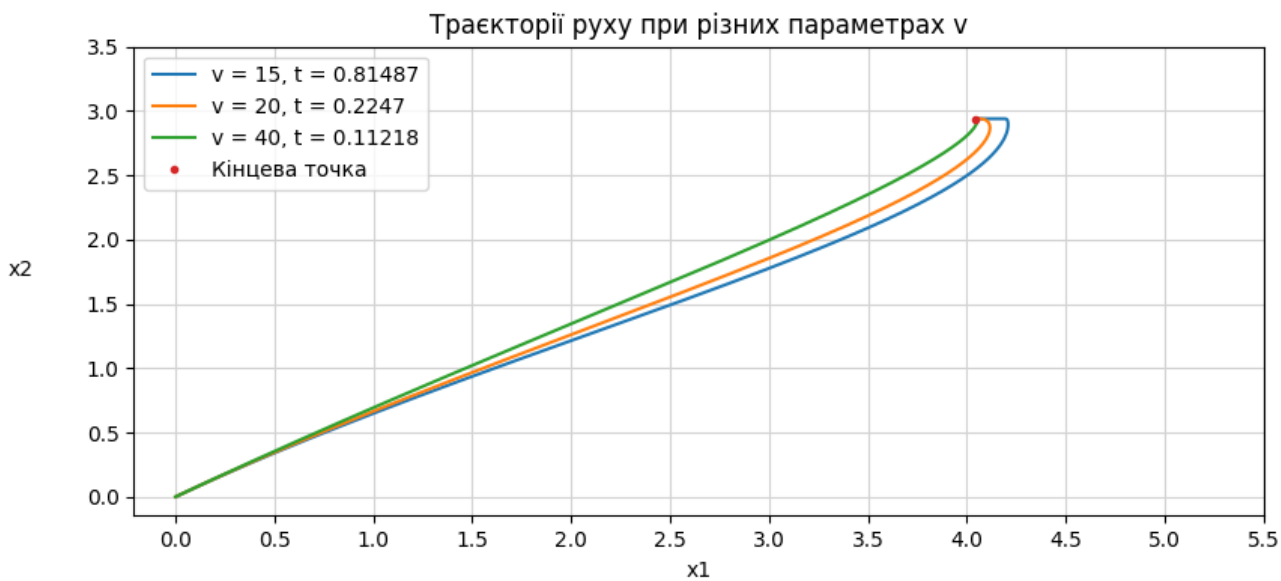


Рис. 8 – траєкторії руху корабля при різних параметрах v

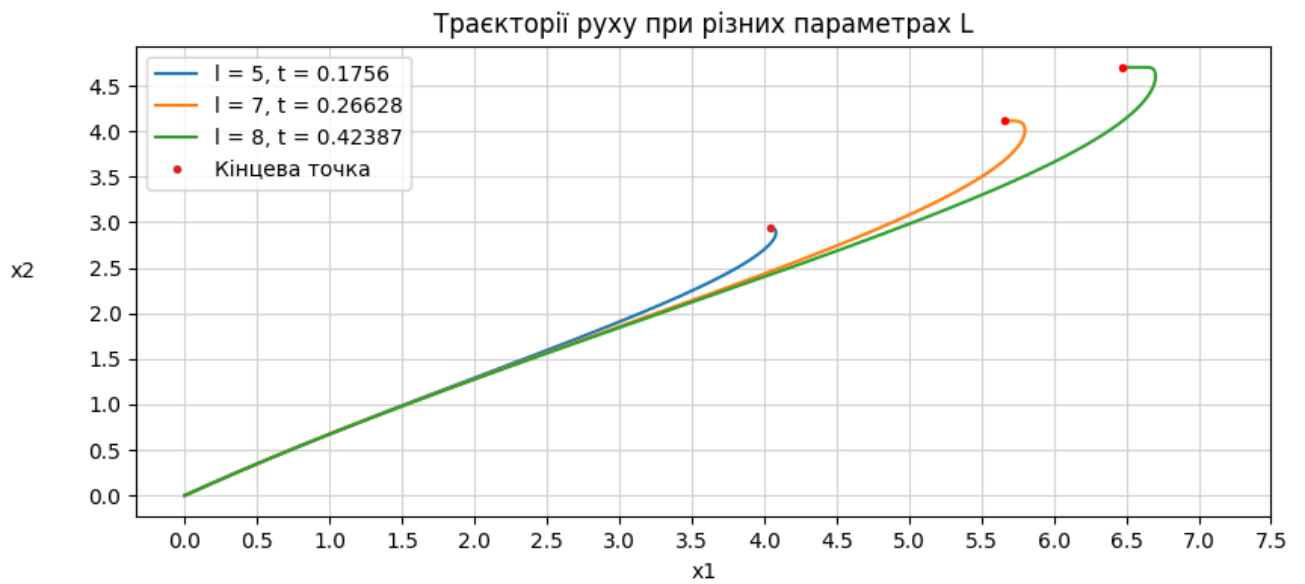


Рис. 9 – траєкторії руху корабля при різних параметрах l

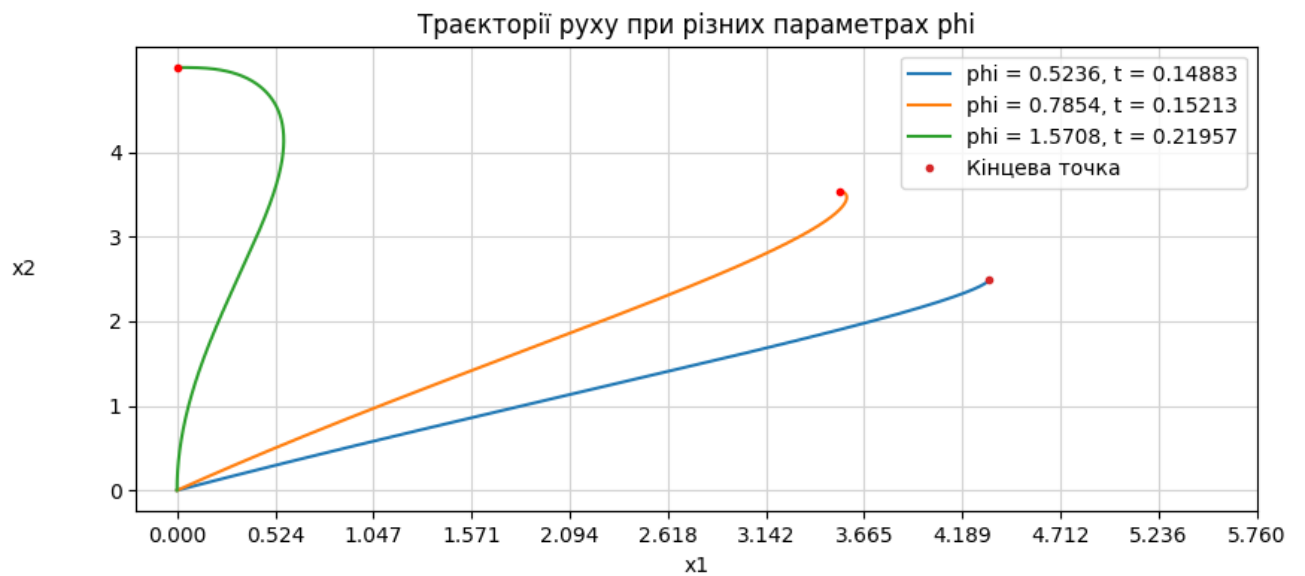


Рис. 10 – траєкторії руху корабля при різних параметрах кута φ

Висновок

У процесі виконанні лабораторної роботи було вивчено принципи побудови траєкторії руху корабля за умови наявності течії та розглянуто вплив різних параметрів на час плавання та вигляд траєкторії:

- Якщо швидкість течії лінійно залежить від координати x_2 , то можна сказати, що підвищення параметру s_0 скорочує час плавання, оскільки течія сприяє руху човна до кінцевої точки. Проте, надмірно високі значення s_0 можуть призвести до того, що корабель пропливе мимо точки.
- Збільшення параметру v дозволяє човну швидше добиратися до кінцевої точки, але при малих значеннях човен зносить течією.
- Збільшення параметру l віддаляє кінцеву точку і збільшує тривалість руху.
- Зміна параметру φ не надто змінює час плавання, однак при виборі кута, близького до $\frac{\pi}{2}$ або $\frac{3\pi}{2}$ корабель може пропливти повз кінцеву точку через значну швидкість течії.
- Зміна параметру N не впливає на те, чи допливе корабель до кінцевої точки та майже не впливає час руху.

Додаток А – Код програми

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

VARIANT:int = 16
s0 = np.sqrt(VARIANT)
# v = np.sqrt(VARIANT)
v = 100
l = VARIANT
phi = VARIANT*np.pi/25
N = 5000

def sx(s, x2):
    return s * x2

def test_s0():
    s0 = np.arange(0.5, 10, 0.5)
    v = 100
    l = VARIANT
    phi = VARIANT*np.pi/25
    N = 5000
    res = []
    for s in s0:
        res.append(calculate(s, v, l, phi, N)[0])
    col = []
    labels = ['Корабель досяг кінцевої точки', 'Корабель проплив кінцеву точку']
    for item in res:
        if item == 0:
            col.append('red')
        else:
            col.append('blue')

    for i in range(len(s0)):
        plt.scatter(s0[i], res[i], c=col[i])
    plt.title('Дослідження параметру s0')
    plt.grid(c='lightgrey', axis='y')
    plt.xlabel("s0")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("t", rotation=0)
```

```

plt.legend(labels, loc='best')
ax = plt.gca()
leg = ax.get_legend()
leg.legend_handles[1].set_color('red')
plt.xticks(np.arange(0.5, 10, 0.5))
plt.yticks(np.arange(0, 1, 0.1))
plt.tight_layout()
plt.show()

def test_v():
    s0 = 5
    v = np.arange(5, 105, 5)
    l = VARIANT
    phi = VARIANT*np.pi/25
    N = 5000
    res = []
    for v0 in v:
        res.append(calculate(s0, v0, l, phi, N)[0])
    col = []
    labels = ['Корабель досяг кінцевої точки', 'Корабель проплив кінцеву точку']
    for item in res:
        if item == 0:
            col.append('red')

        else:
            col.append('blue')

    for i in range(len(v)):
        plt.scatter(v[i], res[i], c=col[i])
    plt.title('Дослідження параметру v')
    plt.grid(c='lightgrey', axis='y')
    plt.xlabel("v")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("t", rotation=0)
    plt.legend(labels, loc='best')
    ax = plt.gca()
    leg = ax.get_legend()
    leg.legend_handles[0].set_color('blue')
    plt.xticks(np.arange(0, 105, 5))
    plt.yticks(np.arange(0, 2, 0.5))
    plt.tight_layout()
    plt.show()

def test_l():
    s0 = 5

```

```

v = 100
l = np.arange(1, 31, 1)
phi = VARIANT*np.pi/25
N = 5000
res = []
for l0 in l:
    res.append(calculate(s0, v, l0, phi, N)[0])
col = []
labels = ['Корабель досяг кінцевої точки', 'Корабель проплив кінцеву точку']
for item in res:
    if item == 0:
        col.append('red')

    else:
        col.append('blue')

for i in range(len(l)):
    plt.scatter(l[i], res[i], c=col[i])
plt.title('Дослідження параметру L')
plt.grid(c='lightgrey')
plt.xlabel("L")
plt.gca().set_ylabel("t", rotation=0)
plt.legend(labels, loc='center left')
ax = plt.gca()
ax.set_axisbelow(True)
leg = ax.get_legend()
leg.legend_handles[1].set_color('red')
plt.xticks(np.arange(0, 31, 1))
plt.yticks(np.arange(0, 14.1, 1))
plt.show()

def test_phi():
    s0 = 5
    v = 60
    l = VARIANT
    phi = np.arange(0, 2*np.pi, np.pi/12)
    N = 5000
    res = []
    for phi0 in phi:
        res.append(calculate(s0, v, l, phi0, N)[0])
    col = []
    labels = ['Корабель досяг кінцевої точки', 'Корабель проплив кінцеву точку']
    for item in res:
        if item == 0:
            col.append('red')

```



```

        else:
            col.append('blue')

    for i in range(len(phi)):
        plt.scatter(phi[i], res[i], c=col[i])
    plt.title('Дослідження параметру phi')
    plt.grid(c='lightgrey')
    plt.xlabel("phi")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("t", rotation=0)
    plt.legend(labels, loc='best')
    ax = plt.gca()
    ax.set_axisbelow(True)
    leg = ax.get_legend()
    leg.legend_handles[1].set_color('red')
    plt.xticks(np.arange(0, 2*np.pi, np.pi/6))
    plt.yticks(np.arange(0, 2.2, 0.2))
    plt.tight_layout()
    plt.show()

def test_n():
    s0 = 5
    v = 100
    l = VARIANT
    phi = VARIANT*np.pi/25
    N = np.arange(1000, 10000, 1000)
    res = []
    for n0 in N:
        res.append(calculate(s0, v, l, phi, n0)[0])
    col = []
    labels = ['Корабель досяг кінцевої точки', 'Корабель проплив кінцеву точку']
    for item in res:
        if item == 0:
            col.append('red')

        else:
            col.append('blue')

    for i in range(len(N)):
        plt.scatter(N[i], res[i], c=col[i])
    plt.title('Дослідження параметру N')
    plt.grid(c='lightgrey')
    plt.xlabel("N")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("t", rotation=0)
    plt.legend(labels, loc='best')

```

```

ax = plt.gca()
ax.set_axisbelow(True)
leg = ax.get_legend()
leg.legend_handles[1].set_color('red')
plt.xticks(np.arange(0, 11000, 1000))
plt.yticks(np.arange(0, 1, 0.1))
plt.tight_layout()
plt.show()

def create_plots():
    plt.subplot(2, 2, 1)
    s0 = [10, 20, 30]
    v = 100
    l = 5
    phi = 5 * np.pi / 25
    N = 5000
    res = []
    time = []
    for s in s0:
        temp = calculate(s, v, l, phi, N)
        res.append([temp[1], temp[2]])
        time.append(temp[0])
    for i in range(len(s0)):
        plt.plot(res[i][0], res[i][1], label='s0 = {}, t = {}'.format(s0[i],
round(time[i], 5)))
    plt.plot(res[0][0][-1], res[0][1][-1], '.', label='Кінцева точка')
    plt.title('Траєкторії руху при різних параметрах s0')
    plt.grid(c='lightgrey')
    plt.xlabel("x1")
    plt.gca().yaxis.set_label_coords(-0.1, 0.5)
    plt.gca().set_ylabel("x2", rotation=0)
    plt.legend(loc='best')
    ax = plt.gca()
    ax.set_axisbelow(True)
    plt.xticks(np.arange(0.5, 6, 0.5))
    plt.yticks(np.arange(0, 5, 1))

    plt.subplot(2, 2, 2)
    s0 = 5
    v = [15, 20, 40]
    l = 5
    phi = 5 * np.pi / 25
    N = 5000
    res = []
    time = []
    for v0 in v:

```

```

        temp = calculate(s0, v0, l, phi, N)
        res.append([temp[1], temp[2]])
        time.append(temp[0])
    for i in range(len(v)):
        plt.plot(res[i][0], res[i][1], label='v = {}, t = {}'.format(v[i],
round(time[i], 5)))
    plt.plot(res[0][0][-1], res[0][1][-1], '.', label='Кінцева точка')
    plt.title('Траєкторії руху при різних параметрах v')
    plt.grid(c='lightgrey')
    plt.xlabel("x1")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("x2", rotation=0)
    plt.legend(loc='best')
    ax.set_axisbelow(True)
    # print(res[0][0])
    plt.xticks(np.arange(0, 6, 0.5))
    plt.yticks(np.arange(0, 4, 0.5))

plt.subplot(2, 2, 3)
s0 = 5
v = 25
l = [5, 7, 8]
phi = 5 * np.pi / 25
N = 5000
res = []
time = []
for l0 in l:
    temp = calculate(s0, v, l0, phi, N)
    res.append([temp[1], temp[2]])
    time.append(temp[0])
for i in range(len(l)):
    plt.plot(res[i][0], res[i][1], label='l = {}, t = {}'.format(l[i],
round(time[i], 5)))
    plt.plot(res[0][0][-1], res[0][1][-1], '.', label='Кінцева точка')
    plt.plot(res[1][0][-1], res[1][1][-1], '.', c='r')
    plt.plot(res[2][0][-1], res[2][1][-1], '.', c='r')
    plt.title('Траєкторії руху при різних параметрах L')
    plt.grid(c='lightgrey')
    plt.xlabel("x1")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("x2", rotation=0)
    plt.legend(loc='best')
    ax.set_axisbelow(True)
    plt.xticks(np.arange(0, 8, 0.5))
    plt.yticks(np.arange(0, 5, 0.5))

plt.subplot(2, 2, 4)

```

```

s0 = 4
v = 30
l = 5
phi = [np.pi/6, np.pi/4, np.pi/2]
rphi = [round(phi[i], 5) for i in range(len(phi))]
N = 5000
res = []
time = []
for phi0 in phi:
    temp = calculate(s0, v, l, phi0, N)
    res.append([temp[1], temp[2]])
    time.append(temp[0])
for i in range(len(phi)):
    plt.plot(res[i][0], res[i][1], label='phi = {}, t = {}'.format(rphi[i],
round(time[i], 5)))
plt.plot(res[0][0][-1], res[0][1][-1], '.', label='Кінцева точка')
plt.plot(res[1][0][-1], res[1][1][-1], '.', c='r')
plt.plot(res[2][0][-1], res[2][1][-1], '.', c='r')
plt.title('Траєкторії руху при різних параметрах phi')
plt.grid(c='lightgrey')
plt.xlabel("x1")
plt.gca().yaxis.set_label_coords(-0.1,0.5)
plt.gca().set_ylabel("x2", rotation=0)
plt.legend(loc='best')
ax.set_axisbelow(True)
plt.xticks(np.arange(0, 2 * np.pi, np.pi / 6))
plt.yticks(np.arange(0, 5, 1))
plt.tight_layout()
plt.show()

def calculate(s0, v, l, phi, N):
    x1_dest = np.array([l * np.cos(phi), l * np.sin(phi)])
    tau = l / (N * v)
    epsilon = 0.001
    x1 = [0]
    x2 = [0]
    k = 0
    critical = np.sqrt((x1[0] - x1_dest[0]) ** 2 + (x2[0] - x1_dest[1]) ** 2)
    while np.sqrt((x1[k] - x1_dest[0]) ** 2 + (x2[k] - x1_dest[1]) ** 2) > epsilon:
        lambd = np.sqrt((x1_dest[0] - x1[k] - sx(s0, x2[k]) * tau) ** 2 +
(x1_dest[1] - x2[k]) ** 2) * v * tau
        u1 = v * tau * (x1_dest[0] - x1[k] - sx(s0, x2[k]) * tau) / lambd
        u2 = v * tau * (x1_dest[1] - x2[k]) / lambd
        x1_temp = x1[k] + (sx(s0, x2[k]) + v * u1) * tau
        x2_temp = x2[k] + v * u2 * tau

```

```

        if np.sqrt((x1_temp - x1_dest[0]) ** 2 + (x2_temp - x1_dest[1]) ** 2) >
critical:
        # print("Корабель проплив кінцеву точку")
        return [0, 0, 0]
        x1.append(x1_temp)
        x2.append(x2_temp)
        k += 1
        # plot_graph(x1, x2, x1_dest, tau, k, s0, v, l, phi, N)
        return tau * k, x1, x2

def plot_graph(x1, x2, x1_fin, tau, k, s0, v, l, phi, N):
    plt.plot(x1, x2, label='Траєкторія руху корабля')
    plt.plot(x1_fin[0], x1_fin[1], '.', label='Кінцева точка')
    plt.title('Навігаційна задача швидкодії\nЧас досягнення: {} | s0={} | v={}
|\n l={} | phi={} | N={}'
              .format(round(tau * k, 5), s0, v, l, round(phi, 5), N))
    plt.grid(c='lightgrey')
    plt.xlabel("x1")
    plt.gca().yaxis.set_label_coords(-0.1,0.5)
    plt.gca().set_ylabel("x2", rotation=0)
    plt.legend(loc='best')
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    calculate(s0, v, l, phi, N)
    test_s0()
    test_v()
    test_l()
    test_phi()
    test_n()
    create_plots()

```