Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 8

з дисципліни «Спеціальні розділи математики-2.  
Чисельні методи»

Виконав:

студент гр. ІС-34

Колосов Ігор

Викладач:

доц. Рибачук Л.В.

Київ – 2024

Тема: Розв’язання задачі Коші

Постановка задачі:

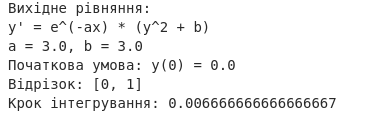
Методами Рунге-Кутта та Адамса розв'язати задачу Коші. На початку інтервалу у необхідній кількості точок значення для методу Адамса визначити методом Рунге-Кутта четвертого порядку.

Для фіксованого h потрібно навести:

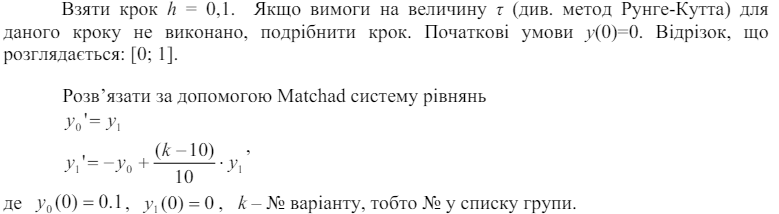
* значення наближеного розв'язку y(x) у тих самих точках, одержані обома методами;
* значення функції помилки ε(x) для обох методів;
* графіки:
  + обох наближених – на одному малюнку;
  + обох помилок – на другому малюнку.

Розв’язати задане рівняння за допомогою Matchad, порівняти із власними результатами.

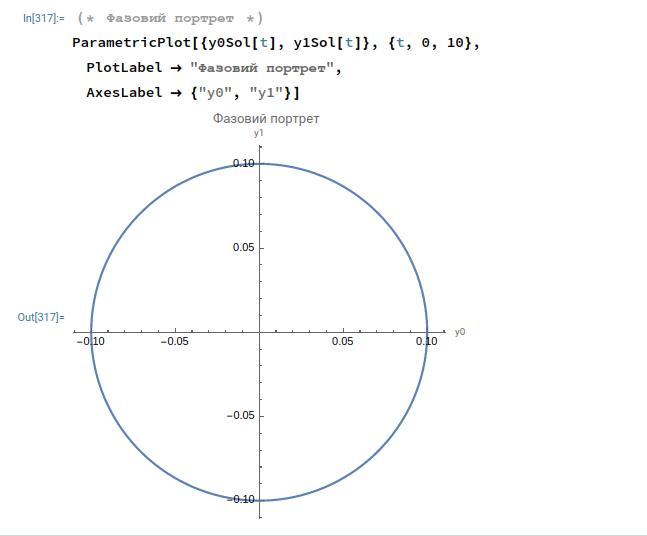
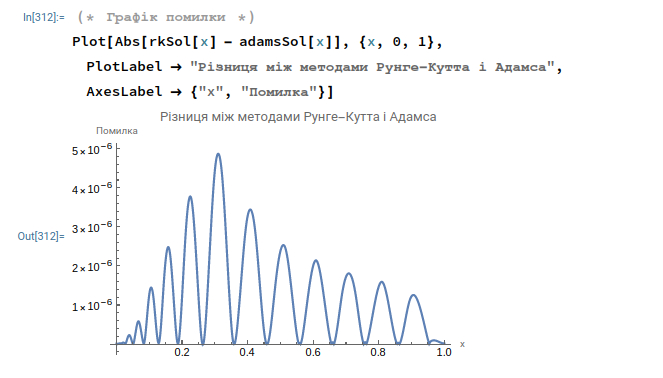
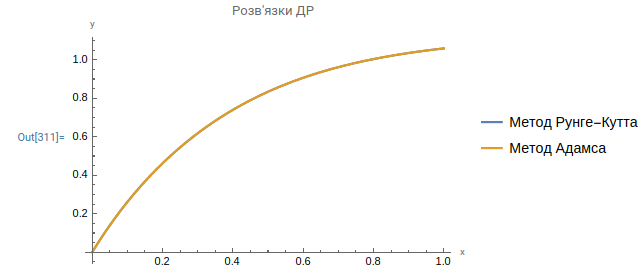
Розв’язати за допомогою Matchad систему рівнянь, побудувати графік y0 та фазовий портрет системи , зробити висновки щодо стійкості системи.



ZZZZZZZ



Розв’язок:



Лістинг програми:

import numpy as np

import matplotlib.pyplot as plt

option = 10

# Метод Рунге-Кутта 4-го порядку

def runge\_kutta(f: callable, x0: float, y0: np.ndarray, xf: float, n: int) -> tuple[np.ndarray, np.ndarray]:

# f - функція правої частини диференціального рівняння y' = f(x, y)

# x0 - початкове значення x

# y0 - початкове значення y (може бути вектором для систем ДР)

# xf - кінцеве значення x

# n - кількість кроків інтегрування

h = (xf - x0) / n # крок інтегрування

x = np.linspace(x0, xf, n + 1) # масив значень x

y = np.zeros((n + 1, len(y0))) # масив для зберігання значень y

y[0] = y0 # початкове значення y

for i in range(n):

# Обчислення коефіцієнтів методу Рунге-Кутта

K1 = f(x[i], y[i]) # K1 = f(x\_i, y\_i)

K2 = f(x[i] + h/2, y[i] + h\*K1/2) # K2 = f(x\_i + h/2, y\_i + h\*K1/2)

K3 = f(x[i] + h/2, y[i] + h\*K2/2) # K3 = f(x\_i + h/2, y\_i + h\*K2/2)

K4 = f(x[i] + h, y[i] + h\*K3) # K4 = f(x\_i + h, y\_i + h\*K3)

# Обчислення наступного значення y

y[i+1] = y[i] + h/6 \* (K1 + 2\*K2 + 2\*K3 + K4)

# Перевірка умови τ для контролю точності

tau = abs((K2 - K3) / (K1 - K2)) if not np.allclose(K1, K2) else 0

try:

b = (tau <= 0.01)

except: pass

if tau.shape == (1, ) and b :

print(f"Умова τ не виконана на кроці {float(i+1)}. Зменшіть крок інтегрування. Tau: ", tau)

return x, y # повертаємо масиви x та y

# Метод Адамса 4-го порядку

def adams(f: callable, x0: float, y0: np.ndarray, xf: float, n: int) -> tuple[np.ndarray, np.ndarray]:

# f - функція правої частини диференціального рівняння y' = f(x, y)

# x0 - початкове значення x

# y0 - початкове значення y (може бути вектором для систем ДР)

# xf - кінцеве значення x

# n - кількість кроків інтегрування

h = (xf - x0) / n # крок інтегрування

x = np.linspace(x0, xf, n + 1) # масив значень x

y = np.zeros((n + 1, len(y0))) # масив для зберігання значень y

# Обчислення перших 4-х значень методом Рунге-Кутта

y[:4] = runge\_kutta(f, x0, y0, x0 + 3\*h, 3)[1]

for i in range(3, n):

# Екстраполяція (прогноз)

y\_pred = y[i] + h/24 \* (55\*f(x[i], y[i]) - 59\*f(x[i-1], y[i-1]) +

37\*f(x[i-2], y[i-2]) - 9\*f(x[i-3], y[i-3]))

# Інтерполяція (корекція)

y\_corr = y[i] + h/24 \* (9\*f(x[i+1], y\_pred) + 19\*f(x[i], y[i]) -

5\*f(x[i-1], y[i-1]) + f(x[i-2], y[i-2]))

# Перевірка умови точності

assert np.allclose(y\_pred, y\_corr, rtol=1e-6), f"Умова точності не виконана на кроці {i+1}. Зменшіть крок інтегрування."

y[i+1] = y\_corr # Зберігаємо скореговане значення

return x, y # повертаємо масиви x та y

# Функція нев'язки

def discrepancy(f: callable, x: np.ndarray, y: np.ndarray, h: float) -> np.ndarray:

# f - функція правої частини диференціального рівняння y' = f(x, y)

# x - масив значень x

# y - масив значень y

# h - крок інтегрування

n = len(x) - 1

res = np.zeros\_like(y) # масив для зберігання значень нев'язки

for i in range(n):

# Обчислення коефіцієнтів методу Рунге-Кутта

K1 = f(x[i], y[i])

K2 = f(x[i] + h/2, y[i] + h\*K1/2)

K3 = f(x[i] + h/2, y[i] + h\*K2/2)

K4 = f(x[i] + h, y[i] + h\*K3)

# Обчислення нев'язки

res[i] = h/6 \* (K1 + 2\*K2 + 2\*K3 + K4) - (y[i+1] - y[i])

res[n] = res[n-1] # екстраполяція для останньої точки

return res # повертаємо масив значень нев'язки

# Завдання 1

def f1(x: float, y: float) -> float:

a = 1.0 + 0.4 \* (option - 5)

b = 1.0 + 0.4 \* (option - 5)

return np.exp(-a\*x) \* (y\*\*2 + b)

x0, y0 = 0, np.array([0.0])

xf = 1

n = 150

h = (xf - x0) / n

print("Вихідне рівняння:")

print("y' = e^(-ax) \* (y^2 + b)")

print(f"a = {1.0 + 0.4 \* (option - 5)}, b = {1.0 + 0.4 \* (option - 5)}")

print(f"Початкова умова: y({x0}) = {y0[0]}")

print(f"Відрізок: [{x0}, {xf}]")

print(f"Крок інтегрування: {h}")

print()

x\_rk, y\_rk = runge\_kutta(f1, x0, y0, xf, n)

x\_adams, y\_adams = adams(f1, x0, y0, xf, n)

print("Значення наближеного розв'язку y(x) у точках:")

print("Метод Рунге-Кутта:")

for i in range(len(x\_rk)):

print(f"x = {x\_rk[i]:.2f}, y = {y\_rk[i, 0]:.6f}")

print()

print("Метод Адамса:")

for i in range(len(x\_adams)):

print(f"x = {x\_adams[i]:.2f}, y = {y\_adams[i, 0]:.6f}")

print()

# Графіки наближених розв'язків

plt.figure(figsize=(8, 6))

plt.plot(x\_rk, y\_rk[:, 0], label='Метод Рунге-Кутта', ls="--", alpha=0.6)

plt.plot(x\_adams, y\_adams[:, 0], label='Метод Адамса', ls="-.", alpha=0.6)

plt.xlabel('x')

plt.ylabel('y')

plt.title('Наближені розв\'язки')

plt.legend()

plt.grid()

plt.show()

eps\_rk = discrepancy(f1, x\_rk, y\_rk, h)

eps\_adams = discrepancy(f1, x\_adams, y\_adams, h)

print("Значення функції нев'язки ε у точках:")

print("Метод Рунге-Кутта:")

for i in range(len(x\_rk)):

print(f"x = {x\_rk[i]:.2f}, ε = {eps\_rk[i, 0]:.6f}")

print()

print("Метод Адамса:")

for i in range(len(x\_adams)):

print(f"x = {x\_adams[i]:.2f}, ε = {eps\_adams[i, 0]:.6f}")

print()

# Графіки похибок

plt.figure(figsize=(8, 6))

plt.plot(x\_rk, eps\_rk[:, 0], ls="--", alpha=0.6, label='Метод Рунге-Кутта')

plt.plot(x\_adams, eps\_adams[:, 0], ls="-.", alpha=0.6, label='Метод Адамса')

plt.xlabel('x')

plt.ylabel('ε')

plt.title('Нев\'язки наближених розв\'язків')

plt.legend()

plt.grid()

plt.show()

# Завдання 2

def f2(x: float, y: np.ndarray) -> np.ndarray:

y0, y1 = y

return np.array([y1, -y0 + (option - 10)/10 \* y1])

x0, y0 = 0, np.array([0.1, 0.0])

xf = 10

n = 100

h = (xf - x0) / n

print("Система диференціальних рівнянь:")

print("y0' = y1")

print(f"y1' = -y0 + {(option - 10)/10} \* y1")

print(f"Початкові умови: y0(0) = {y0[0]}, y1(0) = {y0[1]}")

print(f"Відрізок: [{x0}, {xf}]")

print(f"Крок інтегрування: {h}")

print()

x\_sys, y\_sys = runge\_kutta(f2, x0, y0, xf, n)

# Графік y0

plt.figure(figsize=(8, 6))

plt.plot(x\_sys, y\_sys[:, 0], label='y0')

plt.xlabel('x')

plt.ylabel('y0')

plt.title('Графік y0')

plt.legend()

plt.grid()

plt.show()

# Фазовий портрет

plt.figure(figsize=(8, 6))

plt.plot(y\_sys[:, 0], y\_sys[:, 1])

plt.xlabel('y0')

plt.ylabel('y1')

plt.title('Фазовий портрет')

plt.grid()

plt.show()

— —

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# Параметри

option = 10

a = 1.0 + 0.4 \* (option - 5)

b = 1.0 + 0.4 \* (option - 5)

def f1(x, y):

return np.exp(-a\*x) \* (y\*\*2 + b)

def f2(t, y):

return [y[1], -y[0] + (option - 10)/10 \* y[1]]

x = np.linspace(0, 1, 101)

sol\_rk = solve\_ivp(f1, [0, 1], [0], t\_eval=x, method='RK45')

sol\_adams = solve\_ivp(f1, [0, 1], [0], t\_eval=x, method='RK23')

# Таблиця значень

print("Таблиця значень:")

for i in range(0, len(x), 10):

print(f"x = {x[i]:.2f}, y\_RK = {sol\_rk.y[0][i]:.6f}, y\_Adams = {sol\_adams.y[0][i]:.6f}")

# Графіки розв'язків

plt.figure(figsize=(10, 6))

plt.plot(x, sol\_rk.y[0], label='Метод Рунге-Кутта (scipy)', ls="-.")

plt.plot(x, sol\_adams.y[0], label='Метод Адамса', ls="--")

plt.xlabel('x')

plt.ylabel('y')

plt.title('Розв\'язки ДР')

plt.legend()

plt.grid(True)

plt.show()

# Графік помилки

error = np.abs(sol\_rk.y[0] - sol\_adams.y[0])

plt.figure(figsize=(10, 6))

plt.plot(x, error)

plt.xlabel('x')

plt.ylabel('Помилка')

plt.title('Різниця між методами Рунге-Кутта і Адамса (scipy)')

plt.grid(True)

plt.show()

# Розв'язання системи ДР

t = np.linspace(0, 10, 1000)

sol\_system = solve\_ivp(f2, [0, 10], [0.1, 0], t\_eval=t)

# Фазовий портрет

plt.figure(figsize=(10, 6))

plt.plot(sol\_system.y[0], sol\_system.y[1])

plt.xlabel('y0')

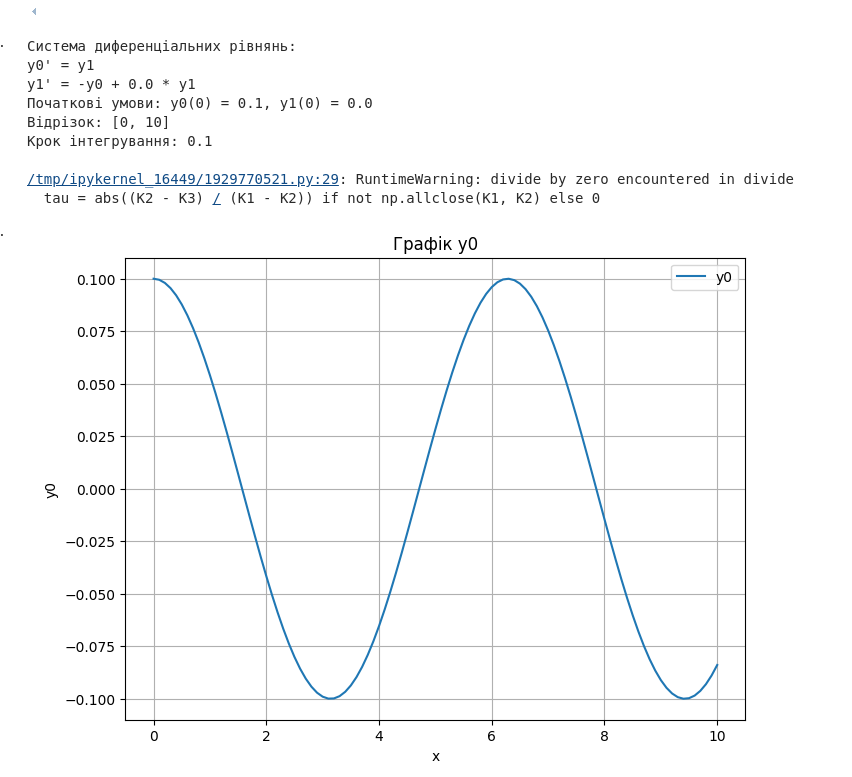
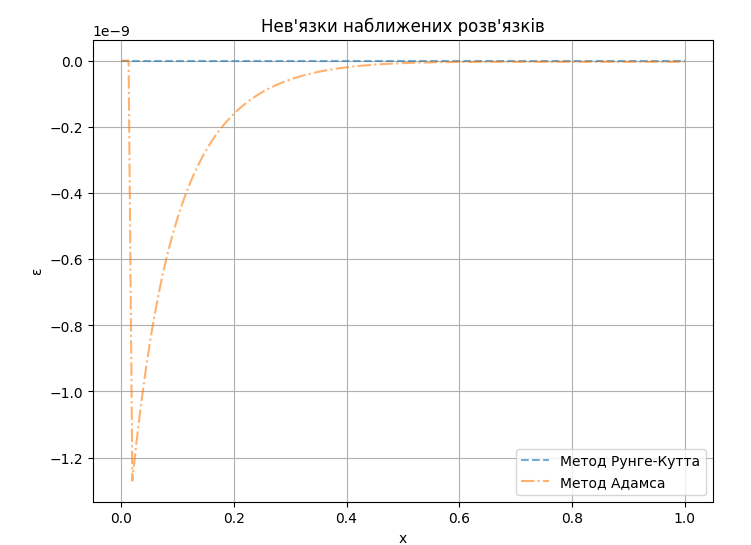
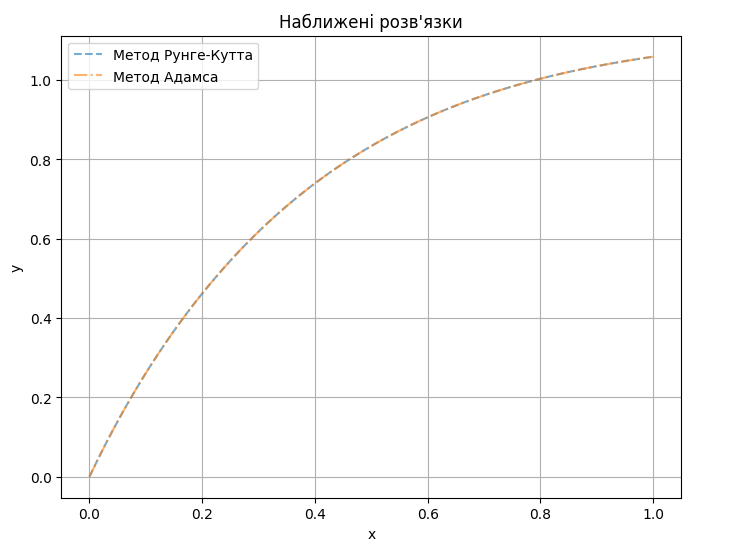
plt.ylabel('y1')

plt.title('Фазовий портрет')

plt.grid(True)

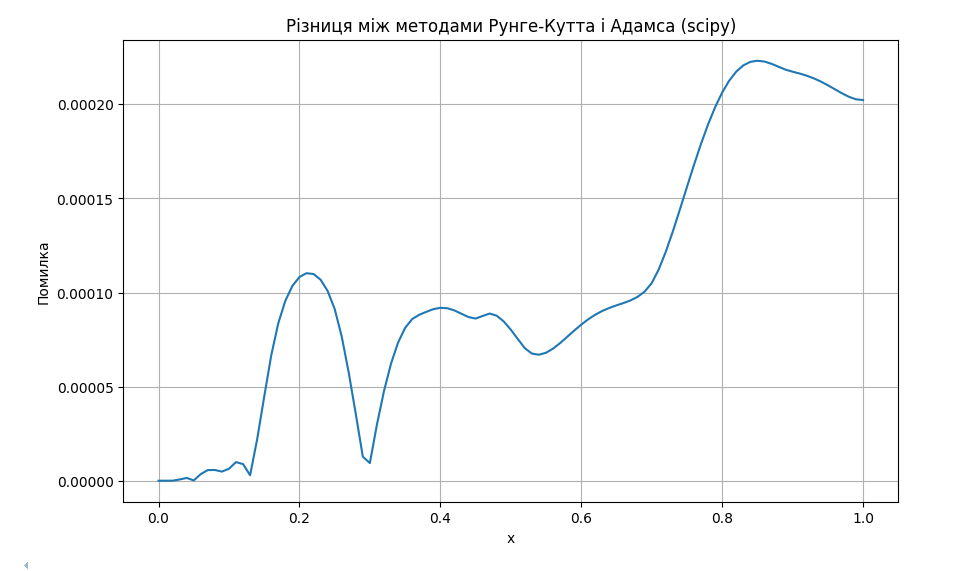
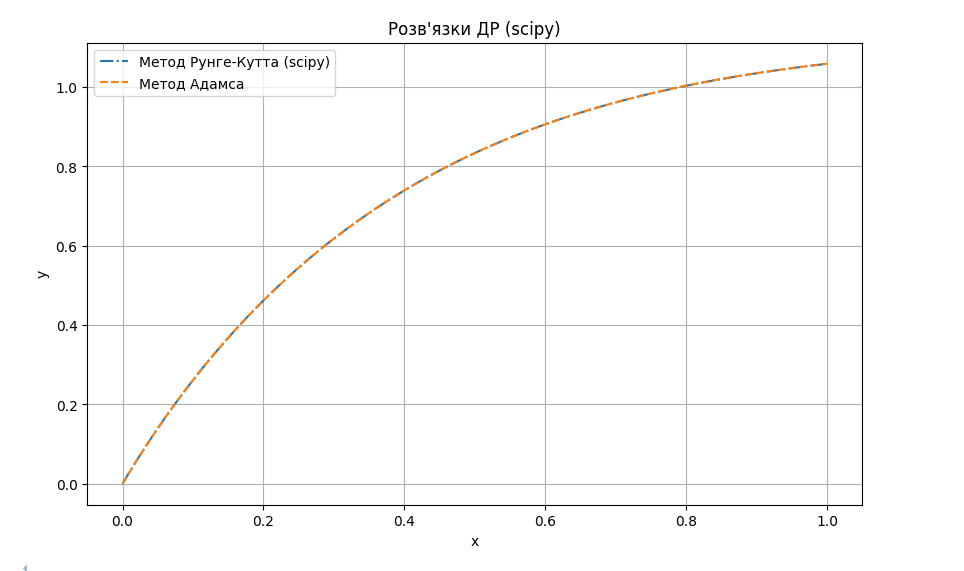
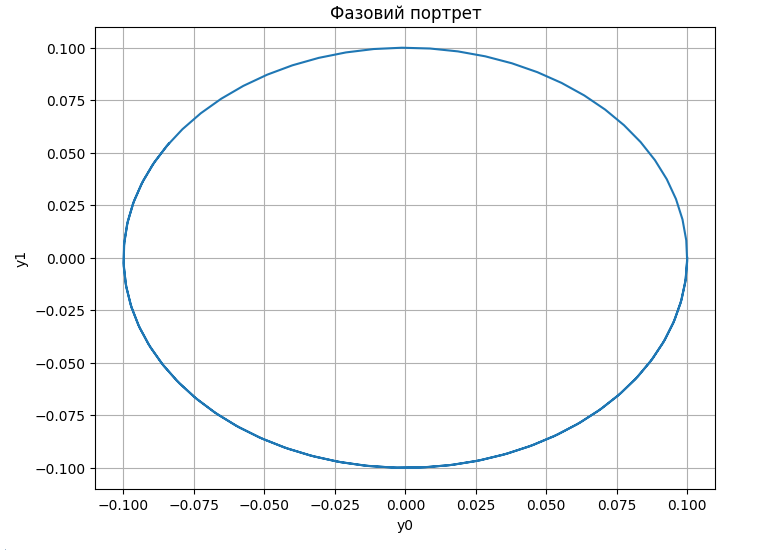
plt.show()

Скріншоти програми



- Фазовий портрет системи диференціальних рівнянь має форму кола, що свідчить про періодичний характер розв'язків.

- Центр кола на фазовому портреті знаходиться в початку координат, що вказує на відсутність зміщення в коливаннях.



Висновок

В цій лабораторній роботі дізнався методи рішення, а також реалізував їх, проблеми Коші. У результаті виявилось, що метод Рунге-Кутта має меншу похибку за метод Адамса.