***Санкт-Петербургский национальный исследовательский университет ИТМО***

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №5

Вариант 9

Выполнила:

Миличенко А.Е.

P3210

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2024 г.

**Цель работы**

Цель лабораторной работы: решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

**Листинг программы**

import numpy as np

import matplotlib.pyplot as plt

import sys

def solution\_1(x, C):

return -(x) - 1 + C \* np.exp(x)

def solution\_2(x, C):

return np.sin(x) / 2 - np.cos(x) / 2 + C / np.exp(x)

def solution\_3(x, C):

return x \*\* 4 / 4 + C

def exact\_solution\_1(y0, x0, xn):

C = y0 + x0 + 1

x = np.linspace(x0, xn, 100000)

y = solution\_1(x, C)

return x, y, lambda x: solution\_1(x, C)

def exact\_solution\_2(y0, x0, xn):

C = np.exp(x0)\*(y0 - np.sin(x0) / 2 + np.cos(x0) / 2)

x = np.linspace(x0, xn, 100000)

y = solution\_2(x, C)

return x, y, lambda x: solution\_2(x, C)

def exact\_solution\_3(y0, x0, xn):

C = y0 - x0 \*\* 4 / 4

x = np.linspace(x0, xn, 100000)

y = solution\_3(x, C)

return x, y, lambda x: solution\_3(x, C)

def cycle(f, y0, i, h, counter):

for j in np.arange(i, i + h + 0.0000000001, h / counter):

y2 = y0 + (h / counter) \* f(j, y0)

if abs(j-i-h) <= 0.001:

return y0

y0 = y2

def euler(f, y0, x0, xn, h, eps):

integer = 0

results = []

popper = 0

R = 0

y0\_prev = y0

p = 1

n = int((xn + 0.001 - x0) / h)

if n > 100000:

print("Слишком большое n, выберете другие интервалы")

sys.exit(0)

for i in np.arange(x0, xn + 0.001, h):

y1 = y0 + h \* f(i, y0)

y2 = y0

counter = 2

if i != x0:

y2 = cycle(f, y0, i - h, h, counter)

R = np.abs(y2 - y0) / (2 \*\* p - 1)

while R > eps:

popper += 1

counter = counter \* 2

y2 = cycle(f, y0, i - h, h, counter)

R = np.abs(y2 - y0) / (2 \*\* counter - 1)

y0 = y2

results.append((i, y2))

integer += 1

y0 = y1

return results, R

def getK(f, i, h, y0):

k1 = h \* f(i, y0)

k2 = h \* f(i + h / 2, y0 + k1 / 2)

k3 = h \* f(i + h / 2, y0 + k2 / 2)

k4 = h \* f(i + h, y0 + k3)

return k1, k2, k3, k4

def runge\_kutta\_4(f, y0, x0, xn, h, eps):

def cycle(y0, i, counter):

for j in np.arange(i, i + h + 0.0000001, h / counter):

k1, k2, k3, k4 = getK(f, j, h / counter, y0)

y2 = y0 + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)

if abs(j-i-h) <= 0.001:

return y0

y0 = y2

integer = 0

results = []

result = []

x0 = x0

y0 = y0

y0\_prev = y0

p = 1

n = int((xn + 0.001 - x0) / h)

if n > 100000:

print("Слишком большое n, выберете другие интервалы")

sys.exit(0)

for i in np.arange(x0, xn + 0.001, h):

k1, k2, k3, k4 = getK(f, i, h, y0)

y1 = y0 + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)

y2 = y0

counter = 1

if i != x0:

counter = 2

y2 = cycle(y0, i - h, counter)

R = np.abs(y2 - y0) / (2 \*\* p - 1)

while R > eps:

counter = counter \* 2

y2 = cycle(y0, i - h, counter)

R = np.abs(y2 - y0) / (2 \*\* counter - 1)

y0 = y2

p = max(p, counter)

result.append((y2, f(i, y2)))

results.append((i, y2))

integer += 1

y0 = y1

return result, results, R

def miln(f, result, h, x0, xn, eps):

y = [result[i][0] for i in range(4)]

f\_vals = [result[i][1] for i in range(4)]

results = [(x0 + h \* i, y[i]) for i in range(4)]

counter = 4

for i in np.arange(x0 + h \* 4, xn + 0.001, h):

y\_prog = y[counter - 4] + 4 \* h / 3 \* (2 \* f\_vals[counter-3] - f\_vals[counter-2] + 2 \* f\_vals[counter - 1])

f\_prog = f(i, y\_prog)

y\_corr = y[counter - 2] + h / 3 \* (f\_vals[counter - 2] + 4 \* f\_vals[counter - 1] + f\_prog)

f\_corr = f(i, y\_corr)

if (i != x0 + h \* 4):

while (abs(y\_corr - y\_prog) >= eps):

y\_prog = y\_corr

f\_prog = f(i, y\_prog)

y\_corr = y[counter - 2] + h / 3 \* (f\_vals[counter - 2] + 4 \* f\_vals[counter - 1] + f\_prog)

f\_corr = f(i, y\_corr)

y.append(y\_corr)

f\_vals.append(f\_corr)

results.append((i, y\_corr))

counter += 1

return results

def main():

f1 = lambda x, y: x + y

f2 = lambda x, y: np.sin(x) - y

f3 = lambda x, y: x \*\* 3

print("f1 = x + y")

print("f2 = sin(x) - y")

print("f3 = x^3")

func\_input = input("Выберите функцию f1/f2/f3 ")

func = f1

if func\_input == "f1":

func = f1

elif func\_input == "f2":

func = f2

elif func\_input == "f3":

func = f3

else:

main()

return

y0 = float(input("Введите начальное условие y0 = y(x0): "))

x0 = float(input("Введите начало интервала дифференцирования x0: "))

xn = float(input("Введите конец интервала дифференцирования xn: "))

h = float(input("Введите шаг h: "))

epsilon = float(input("Введите погрешность eps:"))

if func == f3:

x, y, exact\_sol = exact\_solution\_3(y0, x0, xn)

elif func == f2:

x, y, exact\_sol = exact\_solution\_2(y0, x0, xn)

else:

x, y, exact\_sol = exact\_solution\_1(y0, x0, xn)

plt.plot(x, y, label='Exact solution')

print("Эйлер: ")

dotsEuler, R = euler(func, y0, x0, xn, h, epsilon)

print(f"Точность метода по правилу Рунге: {R}")

print("\nТаблица для метода Эйлера: ")

print("x \t y \t y\_exact")

for x, y in dotsEuler:

y\_exact = exact\_sol(x)

print(f"{x:.2f} \t {y:.2f} \t {y\_exact:.2f}")

print("Рунге-Кутт: ")

result, dotsRunge\_kutt, R = runge\_kutta\_4(func, y0, x0, xn, h, epsilon)

print(f"Точность метода по правилу Рунге: {R}")

print("\nТаблица для метода Рунге-Кутта: ")

print("x \t y \t y\_exact")

for x, y in dotsRunge\_kutt:

y\_exact = exact\_sol(x)

print(f"{x:.2f} \t {y:.2f} \t {y\_exact:.2f}")

dotsMiln = None

if len(dotsRunge\_kutt) >= 4:

print("\nМилн: ")

dotsMiln = miln(func, result, h, x0, xn, epsilon)

print("\nТаблица для метода Милна: ")

print("x \t y \t y\_exact")

for x, y in dotsMiln:

y\_exact = exact\_sol(x)

print(f"{x:.2f} \t {y:.2f} \t {y\_exact:.2f}")

else:

print("Не хватает данных для использования метода Милна")

plt.plot([i[0] for i in dotsEuler], [i[1] for i in dotsEuler], label="Euler")

plt.plot([i[0] for i in dotsRunge\_kutt], [i[1] for i in dotsRunge\_kutt], label="Runge-Kutt")

if dotsMiln is not None:

plt.plot([i[0] for i in dotsMiln], [i[1] for i in dotsMiln], label="Miln")

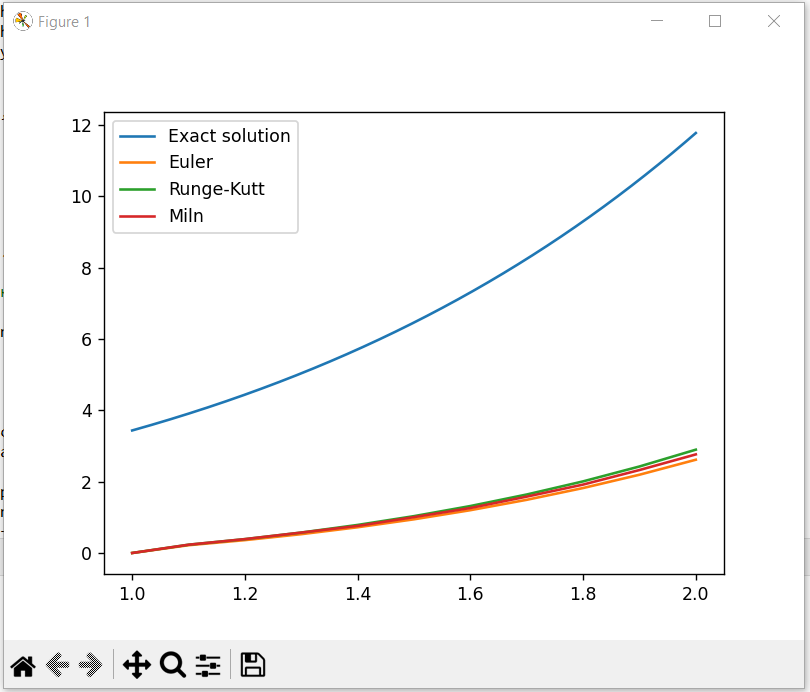
plt.legend()

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы**



f1 = x + y

f2 = sin(x) - y

f3 = x^3

Выберите функцию f1/f2/f3 f1

Введите начальное условие y0 = y(x0): 0

Введите начало интервала дифференцирования x0: 1

Введите конец интервала дифференцирования xn: 2

Введите шаг h: 0.1

Введите погрешность eps:0.1

Эйлер:

Точность метода по правилу Рунге: 0.02854310103847064

Таблица для метода Эйлера:

x y y\_exact

1.00 0.00 3.44

1.10 0.22 3.91

1.20 0.36 4.44

1.30 0.53 5.04

1.40 0.72 5.71

1.50 0.95 6.46

1.60 1.20 7.31

1.70 1.49 8.25

1.80 1.82 9.30

1.90 2.20 10.47

2.00 2.62 11.78

Рунге-Кутт:

Точность метода по правилу Рунге: 0.030750055418787524

Таблица для метода Рунге-Кутта:

x y y\_exact

1.00 0.00 3.44

1.10 0.23 3.91

1.20 0.39 4.44

1.30 0.57 5.04

1.40 0.79 5.71

1.50 1.03 6.46

1.60 1.32 7.31

1.70 1.64 8.25

1.80 2.01 9.30

1.90 2.43 10.47

2.00 2.90 11.78

Милн:

Таблица для метода Милна:

x y y\_exact

1.00 0.00 3.44

1.10 0.23 3.91

1.20 0.39 4.44

1.30 0.57 5.04

1.40 0.76 5.71

1.50 1.01 6.46

1.60 1.26 7.31

1.70 1.58 8.25

1.80 1.92 9.30

1.90 2.33 10.47

2.00 2.77 11.78

**Вывод**

Целью данной лабораторной работы было решить задачу Коши для обыкновенных дифференциальных уравнений (ОДУ) численными методами. В ходе работы мы рассмотрели и применили несколько численных методов, включая метод Эйлера, метод Рунге-Кутты и метод Адамса.