Федеральное государственное автономное образовательное учреждение высшего образования

***Университет ИТМО***

Факультет программной инженерии и компьютерной техники

Дисциплина: **Вычислительная математика**

Лабораторная работа №6

*“* **«Численное дифференцирование»***”*

**Вариант: 9**

Выполнил: Кузнецов Максим Александрович

Группа: Р3111

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург 2022 г

# Цель лабораторной работы: решить задачу Коши численными методами. Написать программную реализацию данных методов по варианту.

# Для исследования использовать:

* Одношаговые методы;
* Многошаговые методы.

**Условия:**

Программная реализация задачи:

1. Исходные данные: ОДУ вида , начальные условия , интервал дифференцирования [*a, b*], шаг *h*, точность .

2. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям. Для оценки точности использовать правило Рунге.

3. Построить графики точного решения и полученного численного решения (разными цветами).

1. Анализ результатов работы: апробация и тестирование.

# Описание методов:

# Методы:

*Одношаговые методы:*

1. Метод Эйлера;
2. Усовершенствованный метод Эйлера;
3. Метод Рунге-Кутта 4- го порядка.

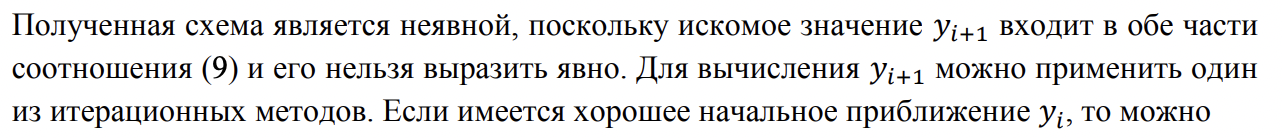
*Многошаговые методы:*

1. Адамса;
2. Милна.

|  |  |  |  |
| --- | --- | --- | --- |
| №  варианта | Метод | №  варианта | Метод |
| 1 | 1, 4 | 16 | 1, 5 |
| 2 | 2, 5 | 17 | 2, 4 |
| 3 | 3, 5 | 18 | 3, 4 |
| 4 | 2, 4 | 19 | 1, 4 |
| 5 | 3, 4 | 20 | 2, 5 |
| 6 | 1, 5 | 21 | 1, 4 |
| 7 | 2, 4 | 22 | 1, 5 |
| 8 | 3, 4 | 23 | 2, 4 |
| 9 | 2, 5 | 24 | 3, 4 |
| 10 | 1, 5 | 25 | 1, 5 |
| 11 | 2, 4 | 26 | 2, 4 |
| 12 | 3, 4 | 27 | 1, 4 |
| 13 | 2, 5 | 28 | 3, 5 |
| 14 | 3, 5 | 29 | 2, 5 |
| 15 | 1, 4 | 30 | 1, 4 |

В программе рассматриваются методы по 9-му варианту: Модифицированный метод Эйлера и метод Милна.

Изображение выглядит как текст, человек, снимок экрана, документ

Автоматически созданное описаниеИзображение выглядит как текст, внутренний, снимок экрана, документ

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

# Листинг программы:

import numpy as np

from matplotlib import pyplot as plt

import math

import sys

import sympy

X, Y = sympy.symbols("x,y")

def show(x, y, desc=""):

plt.plot(x,y,'o')

plt.xlabel("Value of x")

plt.ylabel("Value of y")

plt.title(desc)

plt.show()

# Методом мод. Эйлера

def euler\_modified(eq, a, b, x0, y0, h):

n = int(math.ceil((b - a) / h))

equation = str(eq)

result: list[(float, float)] = [(x0, y0)]

for i in range(n+1):

xi = float(result[i][0])

yi = float(result[i][1])

xi1 = float(xi + h)

f1 = get\_val(equation, xi, yi)

f2 = get\_val(equation, xi1, yi + h \* f1)

yi1 = float(yi + h / 2 \* (f1 + f2))

result.append((xi1, yi1))

return result

# Методом Мильна

def milne(eq, a, b, x0, y0, h, eps):

n: int = math.ceil((b - a) / h)

result, \_, \_ = runge\_rule(eq, a, b, x0, y0, h, eps)

plotX = []

plotYMilne = []

for i in range(3, n+1):

xi = result[i][0]

xi1 = xi + h

y3 = float(result[i - 3][1])

f2 = float(get\_val(eq, result[i - 2][0], result[i - 2][1]))

f1 = get\_val(eq, result[i - 1][0], result[i - 1][1])

fi = get\_val(eq, result[i][0], result[i][1])

y\_cur = float(y3 + 4 \* h / 3 \* (2 \* f2 - f1 + 2 \* fi))

yi1 = float(result[i - 1][1])

fi1r = float(get\_val(eq, xi1, y\_cur))

y\_prev = float(yi1 + h / 3 \* (f1 + 4 \* fi + fi1r))

while abs(y\_prev - y\_cur) > eps:

y\_cur = y\_prev

fi1r = get\_val(eq, xi1, y\_cur)

y\_prev = yi1 + h / 3 \* (f1 + 4 \* fi + fi1r)

result.append((xi1, y\_prev))

return result

def runge\_rule(eq, a, b, x0, y0, h, eps):

eps\_cur = eps

h\_cur = h

res = euler\_modified(eq, a, b, x0, y0, h)

r = 10000

while r >= eps\_cur:

prev\_res = res

h /= 2

res = euler\_modified(eq, a, b, x0, y0, h)

r = abs(res[0][1] - prev\_res[0][1]) / 3

for i in range(len(res[::2])):

r = max(r, abs(res[::2][i][1] - prev\_res[i][1]) / 3)

result = [i for i in res[::int(h\_cur / h)]]

h, h\_cur = h\_cur, h

return result, r, h\_cur

def estimate(y\_arr, y\_arr2, p):

print("Оценка погрешности:")

print(runge\_rule(y\_arr, y\_arr2, p))

# Преобразуем величины в sympy объект

def get\_val(equation: str, x: float, y: float):

return float(sympy.sympify(equation).evalf(subs={X: x, Y: y}))

def resize(array, new\_size, new\_value=0):

"""Resize to biggest or lesser size."""

element\_size = len(array[0]) #Quantity of new elements equals to quantity of first element

if new\_size > len(array):

new\_size = new\_size - 1

while len(array)<=new\_size:

n = tuple(new\_value for i in range(element\_size))

array.append(n)

else:

array = array[:new\_size]

return array

def run():

print("Ур-ие:")

while True:

try:

equation: str = input("y(x)' = ")

get\_val(equation, 1, 1)

break

except (TypeError):

print("Введите снова!", file=sys.stderr)

eq = equation

print("Интервал дифференцирования (a и b):")

while True:

try:

a, b = map(float, input().strip().split(" "))

break

except ValueError:

print("Попробуйте еще раз!", file=sys.stderr)

if a > b:

a, b = b, a

x0, a1, b1 = a, a, b

while True:

try:

print("Введите y0:")

print(f"y({x0}) = ")

a = float(input().strip())

break

except ValueError:

print("Попробуйте еще раз!", file=sys.stderr)

y0 = a

while True:

try:

print("Введите h:")

print(f"h = ")

a = float(input().strip())

break

except ValueError:

print("Попробуйте еще раз!", file=sys.stderr)

h = a

while True:

try:

print("Введите eps:")

print(f"eps = ")

a = float(input().strip())

break

except ValueError:

print("Попробуйте еще раз!", file=sys.stderr)

eps = a

n: int = math.ceil((b1 - a1) / h)

print(n)

euler\_result, \_, \_ = runge\_rule(eq, a1, b1, x0, y0, h, eps)

milne\_result = milne(eq, a1, b1, x0, y0, h, eps)

euler\_result = euler\_result[:n+1]

milne\_result = milne\_result[:n+1]

print(euler\_result)

print(milne\_result)

plotX = []

plotYEuler = []

plotYMilne = []

for index, tuple in enumerate(euler\_result):

plotX.append(tuple[0])

plotYEuler.append(tuple[1])

for index, tuple in enumerate(milne\_result):

plotYMilne.append(tuple[1])

plt.figure()

plt.plot(plotX, plotYEuler, label="Мод. метод Эйлера")

plt.plot(plotX, plotYMilne, label="Метод Милна")

plt.grid(True)

plt.legend()

plt.show()

# out1 = 0

# out2 = 0

# step = h

# while 1:

# euler\_result = euler\_modified(eq, a, b, x0, y0, step)

# euler\_result\_2 = euler\_modified(eq, a, b, x0, y0, step/2)

# if(runge\_rule(euler\_result[0], euler\_result\_2[0], 3)<=eps):

# out1 = euler\_result\_2

# estimate(euler\_result[0], euler\_result\_2[0], 3)

# break

# step = step/2

# euler\_results = out1

# print(euler\_results)

# while 1:

# milne\_result = milne(eq, a, b, x0, y0, step, eps)

# milne\_result\_2 = milne(eq, a, b, x0, y0, step/2, eps)

# if(runge\_rule(milne\_result[0], milne\_result\_2[0], 3)<=eps):

# out2 = milne\_result\_2

# estimate(milne\_result[0], milne\_result\_2[0], 3)

# break

# step = step/2

# milne\_results = out2

# print(milne\_results)

# while 1:

# euler\_result = euler\_modified(eq, a, b, x0, y0, h)

# euler\_result\_2 = euler\_modified(eq, a, b, x0, y0, h/2)

# if(runge\_rule(euler\_result[0][1], euler\_result\_2[0][1], 3)<=eps):

# break

# step = step/2

# show(x\_euler, y\_euler\_2, py\_euler, "Approximation Solution with Modified Euler's Method",5)

# estimate(y\_euler, y\_euler\_2, 3)

# print("Метод Мильна")

# step = 1

# while 1:

# if(runge\_rule(y\_milne, y\_milne\_2, 3)<=0.001):

# break

# step = step/2

# show(x\_milne, y\_milne, py\_milne, "Approximation Solution with Milne's Method",5)

# estimate(y\_milne, y\_milne\_2, 3)

# Пример работы:

Изображение выглядит как текст

Автоматически созданное описание

***Примечание по коду:***

После ввода оценки точности eps на экран выводится количество точек, к которым будут даны значения (другими словами – длина исходного интервала, выраженная в количестве точек). Затем, в первой строке выводится пара значений Xi и Yi для метода мод. Эйлера, и во второй то же самое, только для метода Милна. В конце – график.

# Вывод:

В результате выполнения лабораторной работы я:

* Познакомился с различными способами численного решения задачи Коши с помощью методов мод. Эйлера и Милна
* Написал программную реализацию данных методов на языке Python