ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет программной инженерии и компьютерной техники

Дисциплина:

«Вычислительная математика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Вариант 10.

Выполнил:

Студент гр. P32151

Понамарев Степан Андреевич

Проверил:

Машина Екатерина Алексеевна

Санкт-Петербург, 2023

Цель работы

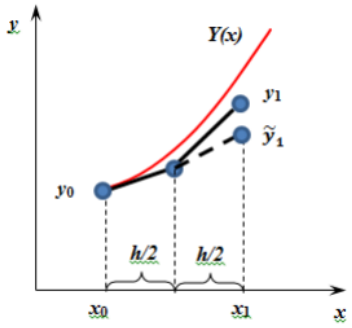
Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Описание алгоритма решения задачи и рабочие формулы

Формула модифицированного метода Эйлера:

$$y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i + hf(x_i, y_i)], i = 0,1, \ldots$$

Данные рекуррентные соотношения описывают новую разностную схему, являющуюся ***модифицированным методом Эйлера***, которая называется методом ***Эйлера с пересчетом***. Метод Эйлера с пересчетом имеет ***второй порядок точности*** $\delta_n = O(h^2)$.



Формула метода Милна:

а) этап прогноза

$$y_i^{\text{прогн}} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

б) этап коррекции

$$y_i^{\text{корр}} = y_{i-2} + \frac{h}{3}\left(f_{i-2} - 4f_{i-1} + 2f_i^{\text{прогн}}\right)$$
$$f_i^{\text{прогн}} = f(x_i, y_i^{\text{прогн}})$$

Для начала счёта требуется задать решения в трёх первых точках, которые можно получить одношаговыми методами (например, методом Рунге-Кутта).

Суммарная погрешность этого метода есть величина $\delta_n = O(h^4)$.

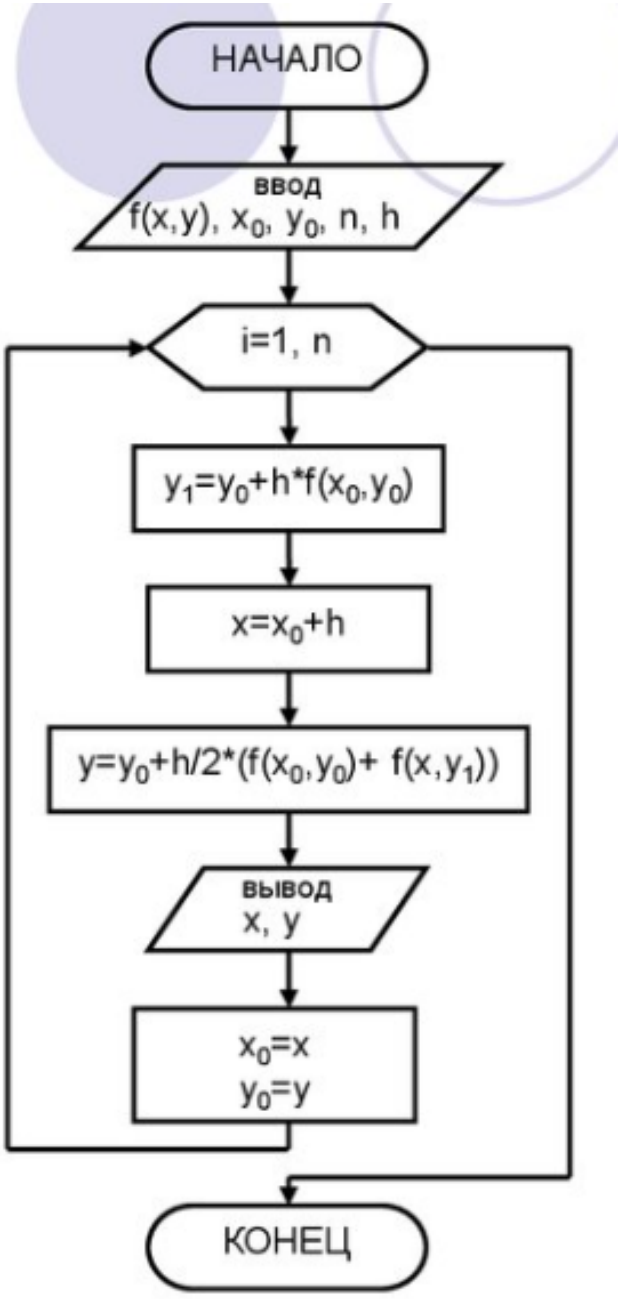
Формула Рунге-Кутта при $k = 4$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

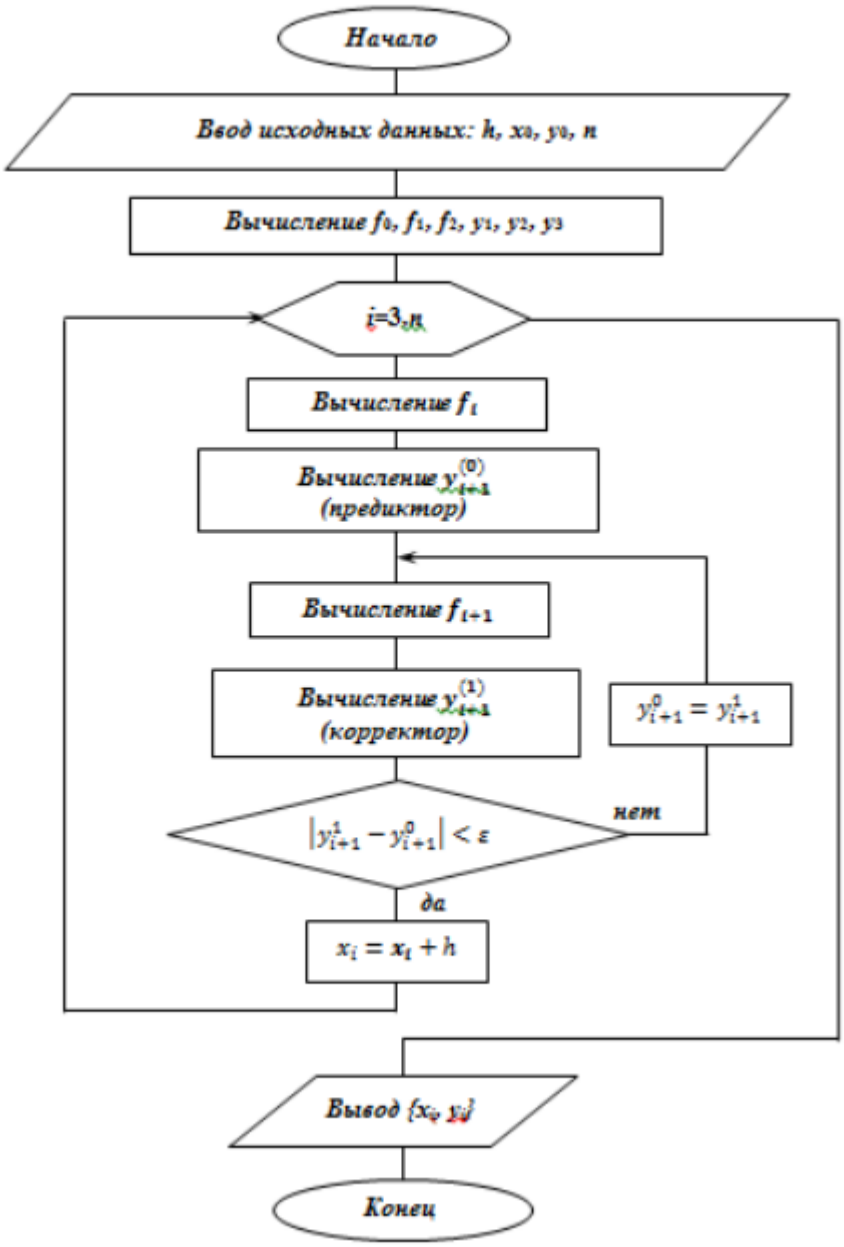
$k_1 = h \cdot f(x_i, y_i)$

$$k_2 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$
$$k_3 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$$
$$k_4 = h \cdot f(x_i + h, y_i + k_3)$$

Блок-схема модифицированного метода Эйлера:



Блок-схема метода Милна:



Листинг программы

Main.py

```
import math

import numpy as np
from matplotlib import pyplot as plt

from Functions import FunctionHolder
from InputManager import InputManager
```

```
def get_n(x0, xn, h):
    return int(math.ceil(round(abs(xn - x0) / h, 3)))


def euler_method(func, x0, xn, y0, h):
    """
    Метод Эйлера
    """
    x_previous = [x0]
    y_previous = [y0]
    n = get_n(x0, xn, h)
    for i in range(n):
        x_now = x_previous[i]
        y_now = y_previous[i]
        x_previous.append(x_now + h)
        y_previous.append(y_now + h * (func(x_now, y_now) +
func(x_now + h, y_now + h * func(x_now, y_now))) / 2)

    return x_previous, y_previous


def runge_kutta_method(func, x0, xn, y0, h):
    """
    Метод Рунге-Кутта 4-го порядка
    """
    x_previous = [x0]
    y_previous = [y0]
    n = get_n(x0, xn, h)
    for i in range(n):
        x_now = x_previous[i]
        y_now = y_previous[i]
        k1 = h * func(x_now, y_now)
        k2 = h * func(x_now + h / 2, y_now + k1 / 2)
        k3 = h * func(x_now + h / 2, y_now + k2 / 2)
        k4 = h * func(x_now + h, y_now + k3)
        x_previous.append(x_now + h)
        y_previous.append(y_now + (k1 + 2 * k2 + 2 * k3 + k4) / 6)
    return x_previous, y_previous


def milne_method(func, x0, xn, y0, h):
    n = get_n(x0, xn, h)
    if n < 5:
        return runge_kutta_method(func, x0, xn, y0, h)
    # y1, y2, y3 считаются методом Рунге-Кутта 4-го порядка
    x_previous, y_previous = runge_kutta_method(func, x0, x0 + 3 *
h, y0, h)

    x_now = x_previous[-1]
    y_now = y_previous[-1]
```

```
    f = lambda i: func(x_previous[i], y_previous[i])
    for i in range(4, n + 1):
        x_now += h
        # этап прогноза
        y_now = y_previous[i - 4] + 4 * h * (2 * f(i - 3) - f(i -
2) + 2 * f(i - 1)) / 3
        # этап коррекции
        y_now = y_previous[i - 2] + h * (f(i - 2) + 4 * f(i - 1) +
func(x_now, y_now)) / 3

        x_previous.append(x_now)
        y_previous.append(y_now)

    return x_previous, y_previous


def get_inputs():
    x0 = InputManager.float_input("Введите левый край отрезка x0 =
")
    xn = InputManager.float_input("Введите правый край отрезка xn =
")
    y0 = InputManager.float_input("Введите y(x0): ")
    h = InputManager.float_input("Введите шаг h = ")
    return x0, xn, y0, h


if __name__ == '__main__':
    """
    y' = e^sin(x) * cos(x), answer: y = e^sin(x) + C
    y' = x + 1, answer = x^2/2 + x + C
    y' = 2xy - x^2 - y^2 + 5, answer: y = 4 / (C * e^4x - 1) + x +
2
    """
    variants = ["y' = e^sin(x) * cos(x)",
                "y' = x + 1"]
    values = [FunctionHolder(lambda x, y: np.exp(np.sin(x)) *
np.cos(x), lambda x, c: np.exp(np.sin(x)) + c),
              FunctionHolder(lambda x, y: x + 1, lambda x, c: x **
2 / 2 + x + c)]
    chosen_func = InputManager.multiple_choice_input(variants,
values, "Выберите ОДУ для численного решения:")

    variants = ["Метод Эйлера", "Метод Рунге-Кутта 4-го порядка",
"Метод Милна"]
    values = [euler_method, runge_kutta_method, milne_method]
    chosen_method = InputManager.multiple_choice_input(variants,
values, "Выберите метод для интегрирования: ")

    x0, xn, y0, h = get_inputs()
    c = y0 - chosen_func.answer(x0, 0)
```

```
    plt.title("Численное решение ОДУ")
    plt.plot(*chosen_method(chosen_func.f, x0, xn, y0, h), c='b')
    x = np.linspace(x0, xn, 50)
    plt.plot(x, chosen_func.answer(x, c), c='r')
    plt.grid(True)
    plt.show()
```

InputManager.py

```
class InputManager:
    @staticmethod
    def string_input(message=""):
        buf = ""
        while buf == "":
            buf = input(message).strip()
        return buf

    @staticmethod
    def _check_number(buf):
        try:
            float(buf.replace(',', '.'))
            return True
        except ValueError:
            return False

    @staticmethod
    def _convert_to_number(num):
        try:
            return float(num.replace(',', '.'))
        except ValueError:
            return None

    @staticmethod
    def float_input(message=""):
        number = None
        while number is None:
            number =
InputManager._convert_to_number(InputManager.string_input(message))
        return number

    @staticmethod
    def int_input(message=""):
        return int(InputManager.float_input(message))

    @staticmethod
    def yes_or_no_input(message=""):
        answer = "0"
        while answer[0].lower() not in ["y", "n", "д", "н"]:
            answer = InputManager.string_input(message + " [y/n]:
")
```

```
        return answer[0].lower() in ["у", "д"]

    @staticmethod
    def enum_input(variants_list, message=""):
        variants_list = [str(i) for i in variants_list]
        buf = ""
        while buf not in variants_list:
            buf = InputManager.string_input(message)
        return buf

    @staticmethod
    def multiple_choice_input(variant_list, values_list,
message=""):
        n = len(variant_list)
        if n == 0 or len(variant_list) != len(values_list):
            raise ValueError

        if message != "":
            print(message)

        for i in range(n):
            s = f"{i + 1}."
            lines = variant_list[i].split('\n')
            print('\t' + s, lines[0])
            for line in lines[1:]:
                print("\t" + ' ' * len(s), line)

        i = int(InputManager.enum_input([*range(1, n + 1)],
f"Введите число от 1 до {n}: "))
        return values_list[i - 1]

    @staticmethod
    def epsilon_input(message=""):
        e = InputManager.float_input(message)
        while not (0 < e and e <= 1):
            print("Эпсилон должно быть в промежутке от 0 до 1!")
            e = InputManager.float_input(message)
        return e

    @staticmethod
    def int_input_with_borders(left, right, message=""):
        if left >= right:
            raise ValueError
        if message != "":
            print(message)
        i = int(InputManager.enum_input([*range(left, right + 1)],
f"Введите число от {left} до {right}: "))
        return i

    @staticmethod
    def point_input(message=""):
```

```
        x, y = None, None
        while x is None or y is None:
            line = InputManager.string_input(message).split()
            if len(line) != 2:
                continue
            x, y = InputManager._convert_to_number(line[0]),
InputManager._convert_to_number(line[1])
        return x, y

    @staticmethod
    def float_input_with_borders(left, right, message=""):
        if not left < right:
            raise ValueError
        x = InputManager.float_input(message)
        while not (left < x < right):
            print(f"Значение должно быть от {left} до {right}!")
            x = InputManager.float_input(message)
        return x
```

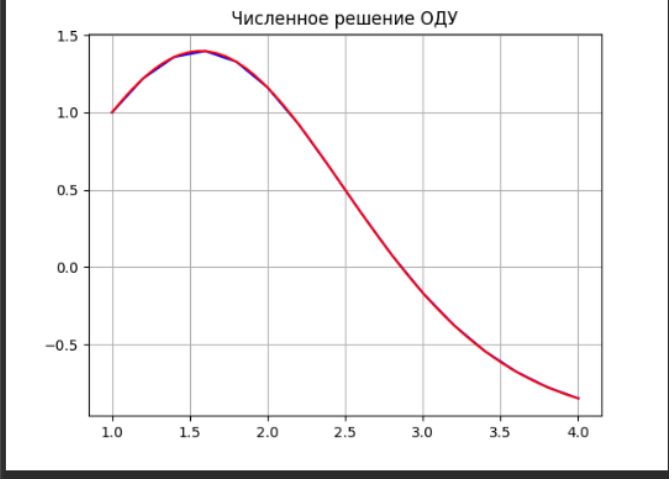
Functions.py

```
class FunctionHolder:
    def __init__(self, derivative, func):
        self.f = derivative
        self.answer = func
```

Примеры и результат работы программы

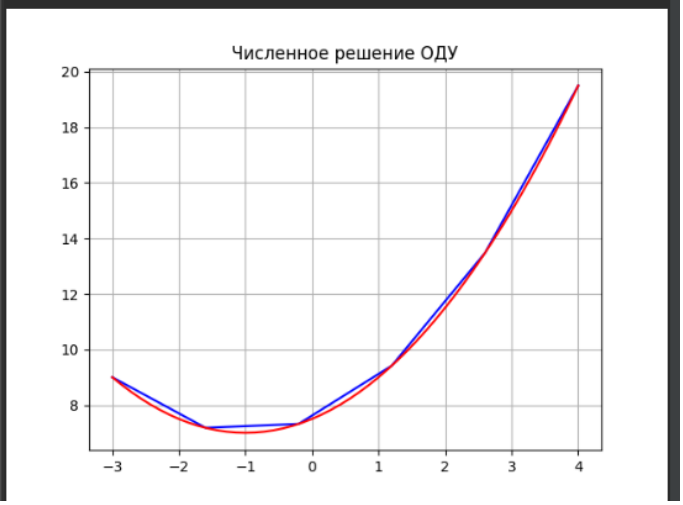
Пример 1

```
Выберите ОДУ для численного решения:
    1. y' = e^sin(x) * cos(x)
    2. y' = x + 1
Введите число от 1 до 2: 1
Выберите метод для интегрирования:
    1. Метод Эйлера
    2. Метод Рунге-Кутта 4-го порядка
    3. Метод Милна
Введите число от 1 до 3: 2
Введите левый край отрезка x0 = 1
Введите правый край отрезка xn = 4
Введите y(x0): 1
Введите шаг h = 0.2
```



Пример 2

```
Выберите ОДУ для численного решения:
    1. y' = e^sin(x) * cos(x)
    2. y' = x + 1
Введите число от 1 до 2: 2
Выберите метод для интегрирования:
    1. Метод Эйлера
    2. Метод Рунге-Кутта 4-го порядка
    3. Метод Милна
Введите число от 1 до 3: 3
Введите левый край отрезка x0 = -3
Введите правый край отрезка xn = 4
Введите y(x0): 9
Введите шаг h = 1.4
```



Выводы

В ходе лабораторной работы я изучил и реализовал на языке Python 3 несколько методов численного решения дифференциальных уравнений.