Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №6

по дисциплине

«Вычислительная математика»

Вариант №13

Выполнил:

Студент группы P3213

Султанов А.Р.

Проверила:

Машина Е.А.

г. Санкт-Петербург

2024г.

Цель работы

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Задание

1. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;

2. Пользователь выбирает ОДУ вида $y' = f(x, y)$ (не менее трех уравнений), из тех, которые предлагает программа;

3. Предусмотреть ввод исходных данных с клавиатуры: начальные условия $y_0 = y(x_0)$, интервал дифференцирования $[x_0, x_n]$, шаг h, точность ε;

4. Для исследования использовать одношаговые методы и многошаговые методы (см. табл.1);

5. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям, для всех методов, реализуемых в программе;

6. Для оценки точности одношаговых методов использовать правило Рунге;

7. Для оценки точности многошаговых методов использовать точное решение задачи: $\varepsilon = max_{0 \le i \le n} |y_{i\text{точн}} - y_i|$

8. Построить графики точного решения и полученного приближенного решения (разными цветами);

9. Проанализировать результаты работы программы.

Формулы

Метод Эйлера: $y_{i+1} = y_i + hf(x_i, y_i)$

Модифицированный метод Эйлера:

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

Метод Милна:

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

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

$$f_i^{\text{прогн}} = f(x_i, y_i^{\text{прогн}})$$

Пример работы программы

```
python main.py
Выберите метод [1-3]:
1. 3x + 7y
2. y + cos(2x)
3. x^3 + y
2
Введите x0
0
Введите y0
0
Введите xn
3
Введите h
1
Введите точность
0.1
Метод Эйлера
Точность Метод Эйлера по правилу Рунге: 0.09349849
 x | y
0.0 | 0.0
0.125 | 0.125
0.25 | 0.2617390527138306
0.375 | 0.404154254539356
0.5 | 0.5461346449660032
0.625 | 0.681939263820271
0.75 | 0.8065969670972135
0.875 | 0.916263738192828
1.0 | 1.008515948510745
```

1.125 | 1.0825620875061954
1.25 | 1.1393606456041274
1.375 | 1.1816377743612767
1.5 | 1.2138046988273783
1.625 | 1.2417812241057449
1.75 | 1.2727376676088946
1.875 | 1.314772790148657
2.0 | 1.376549469249794
2.125 | 1.4669127002980666
2.25 | 1.5945158515961009
2.375 | 1.767480858116766
2.5 | 1.993116234492359
2.625 | 2.2777135369868073
2.75 | 2.6264384137653884
2.875 | 3.0433269372724694
3.0 | 3.531391856576718

Усовершенствованный метод Эйлера
Точность Усовершенствованный метод Эйлера по правилу Рунге: 0.07679028
 x | y
0.0 | 0.0
0.5 | 0.5100755764670349
1.0 | 0.9274494673226987
1.5 | 1.1035521965440958
2.0 | 1.2586142279430856
2.5 | 1.8710473089494664
3.0 | 3.3868677682541843
3.6572265625 | 8.117173978890435

Метод Милна (пошаговый)
Точность Метод Милна (пошаговый): 0.10000000 79620528853
 x | y
0.0 | 0.0
0.0625 | 0.0625
0.125 | 0.12841860420183307
0.1875 | 0.2762725715882025
0.25 | 0.2760749185025383
0.3125 | 0.42146992511951986
0.375 | 0.42719772159189207
0.4375 | 0.5687193020847315
0.5 | 0.5756264599412381
0.5625 | 0.7104777162117232

0.625 | 0.7157091012654823
0.6875 | 0.8415555485010551
0.75 | 0.8427684438524587
0.8125 | 0.9578925068325438
0.875 | 0.953456106615261
0.9375 | 1.0569075197438775
1.0 | 1.046030587986118
1.0625 | 1.13773258205033
1.125 | 1.120544774560682
1.1875 | 1.2013506417639195
1.25 | 1.1789323914869685
1.3125 | 1.2506295997171648
1.375 | 1.2249894236738652
1.4375 | 1.2902518121304016
1.5 | 1.2642532506895776
1.5625 | 1.3265451683353757
1.625 | 1.3037888093013883
1.6875 | 1.3672281607291645
1.75 | 1.3518971176524437
1.8125 | 1.4210869665353436
1.875 | 1.4177665936136168
1.9375 | 1.497607077031968
2.0 | 1.5110914641298385
2.0625 | 1.6065851632942914
2.125 | 1.6416839569496093
2.1875 | 1.7577484672534804
2.25 | 1.8191077475527622
2.3125 | 1.9604089593294811
2.375 | 2.052359259967374
2.4375 | 2.223177818588885
2.5 | 2.3496209518410867
2.5625 | 2.5537625805432027
2.625 | 2.7181068130059005
2.6875 | 2.958864769417599
2.75 | 3.1640152243903987
2.8125 | 3.444190277100049
2.875 | 3.6925983866728487
2.9375 | 4.0145785262277585
3.0 | 4.308351116236711

Исходный код

Листинг программы доступен по ссылке:

https://github.com/MakeCheerfulInstall/Computational-Math-2024/pull/47

Метод Эйлера:

```
def euler(f: Function, x0: float, y0: float, xn: float, h: float) ->
tuple[Xs, Ys]:
    """Метод Эйлера."""
    n = int((xn - x0) / h)

    xs = [x0 + h * i for i in range(n + 1)]
    ys = [0 for i in range(n + 1)]

    ys[0] = y0

    for i in range(n):
        ys[i + 1] = ys[i] + h * f(xs[i], ys[i])

    return xs, ys
```

Модифицированный метод Эйлера:

```
def improved_euler(f: Function, x0: float, y0: float, xn: float, h: float)
-> tuple[Xs, Ys]:
    """Модифицированный метод Эйлера."""
    n = int((xn - x0) / h)

    xs = [x0 + h * i for i in range(n + 1)]
    ys = [0 for i in range(n + 1)]

    ys[0] = y0

    for i in range(n):
        x_i, y_i = xs[i], ys[i]
        x_i1 = xs[i + 1]
        ys[i + 1] = y_i + (h / 2) * (f(x_i, y_i) + f(x_i1, y_i + h * f(x_i,
y_i)))

    return xs, ys
```

Метод Милна:

```
def milne(f: Function, x0: float, y0: float, xn: float, h: float) ->
tuple[Xs, Ys]:
    """Метод Милна."""
    n = int((xn - x0) / h)

    xs, ys = euler(f, x0, y0, xn, h)

    for i in range(3, n + 1):
        ys[i] = ys[i - 4] + (4 * h / 3) * (2 * f(xs[i - 3], ys[i - 3]) -
f(xs[i - 2], ys[i - 2]) + 2 * f(xs[i - 1], ys[i - 1]))
        ys[i] = ys[i - 2] + (h / 3) * (f(xs[i - 2], ys[i - 2]) + 4 * f(xs[i
- 1], ys[i - 1]) + f(xs[i], ys[i]))

    return xs, ys
```