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

Факультет программной инженерии и компьютерной техники

Вычислительная математика
Лабораторная работа № 6
Вариант: 3

Выполнил:
Вильданов Ильнур Наилевич
Группа №P3212

Проверила:
Машина Екатерина Алексеевна

Санкт-Петербург
2025 г.

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. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных.
10. Проанализировать результаты работы программы.

Рабочие формулы

Метод Эйлера:

$$\boldsymbol{y_{i+1} = y_i + hf(x_i, y_i)}$$

Метод Рунге-Кутта 4-го порядка:

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

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

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

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

$$\boldsymbol{k_4 = h \cdot f(x_i + h, y_i + k_3)}$$

Метод 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:

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

Листинг программы

```
import math

def euler_method(f, x0: float, xn: float, y0: float, h: float) ->
tuple[list[float], list[float]]:
    xs = [x0]
    ys = [y0]
    x = x0
    while x < xn and not math.isclose(x, xn):
        y_prev = ys[-1]
        y_next = y_prev + h * f(x, y_prev)
        x = round(x + h, 12)
        xs.append(x)
        ys.append(y_next)
    return xs, ys
```

```
import math


def rk4_method(f, x0: float, xn: float, y0: float, h: float) ->
tuple[list[float], list[float]]:
    xs = [x0]
    ys = [y0]
    x = x0
    while x < xn and not math.isclose(x, xn):
        y = ys[-1]

        k1 = h * f(x, y)
        k2 = h * f(x + h / 2, y + k1 / 2)
        k3 = h * f(x + h / 2, y + k2 / 2)
        k4 = h * f(x + h, y + k3)

        y_next = y + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        x = round(x + h, 12)
        xs.append(x)
        ys.append(y_next)
    return xs, ys
```

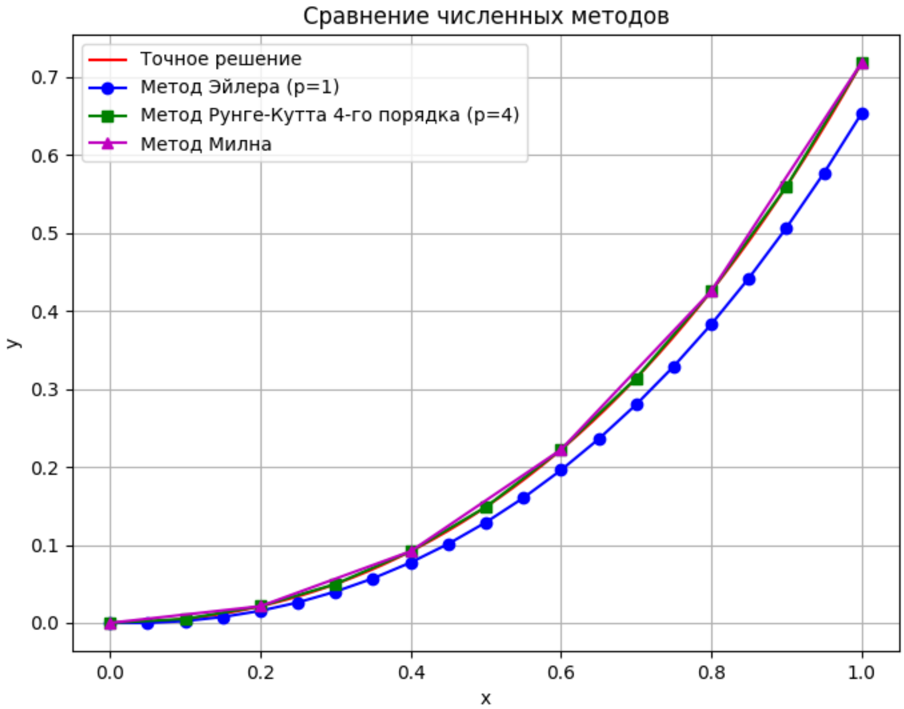
```
import math
from methods.rk4 import rk4_method


def milne_method(f, x0: float, xn: float, y0: float, h: float) ->
tuple[list[float], list[float]]:
    xs_rk, ys_rk = rk4_method(f, x0, x0 + 3*h, y0, h)
    xs = xs_rk.copy()
    ys = ys_rk.copy()

    while xs[-1] < xn and not math.isclose(xs[-1], xn):
```

```python
        i = len(xs) - 1  # текущий индекс
        x_i   = xs[i];    y_i   = ys[i]
        x_im1, y_im1 = xs[i-1], ys[i-1]
        x_im2, y_im2 = xs[i-2], ys[i-2]
        x_im3, y_im3 = xs[i-3], ys[i-3]

        # предиктор
        y_pred = y_im3 + (4*h/3) * (
            2*f(x_im2, y_im2)
            - f(x_im1, y_im1)
            + 2*f(x_i,   y_i)
        )
        x_next = round(x_i + h, 12)

        # корректор
        y_corr = y_im1 + (h/3) * (
            f(x_im1, y_im1)
            + 4 * f(x_i, y_i)
            + f(x_next, y_pred)
        )

        xs.append(x_next)
        ys.append(y_corr)

    return xs, ys
```

Примеры и результаты работы программы

```
=== Лабораторная №6: Численное решение ОДУ ===

1) y' = y/3 + 2x
2) y' = x + y
3) y' = 2y + cos(x)
4) y' = y + (1 + x)·y²
Ваш выбор (1-4): 2
Введите x0 (начало): 0
Введите xn (конец, > x0): 1
Введите y(x0) = y0: 0
Введите начальный шаг h: 0.2
Введите точность ε: 0.1

=== Метод Эйлера (p=1) ===
Достигнута точность ε = 1.00e-01 при эффективном шаге h = 0.05
Погрешность: 0.0595552
   x       y_num       y_true     погрешность
----  ---------  ----------  -------------
0     0          0              0
0.05  0          0.0012711      0.0012711
0.1   0.0025     0.00517092     0.00267092
0.15  0.007625   0.0118342      0.00420924
0.2   0.0155063  0.0214028      0.00589651
0.25  0.0262816  0.0340254      0.00774385
0.3   0.0400956  0.0498588      0.00976317
0.35  0.0571004  0.0690675      0.0119671
0.4   0.0774554  0.0918247      0.0143693
0.45  0.101328   0.118312       0.016984
```

```
0.5   0.128895   0.148721       0.0198266
0.55  0.160339   0.183253       0.0229137
0.6   0.195856   0.222119       0.0262625
0.65  0.235649   0.265541       0.0298917
0.7   0.279932   0.313753       0.0338211
0.75  0.328928   0.367          0.0380718
0.8   0.382875   0.425541       0.0426663
0.85  0.442018   0.489647       0.0476285
0.9   0.506619   0.559603       0.0529839
0.95  0.57695    0.63571        0.0587595
1     0.653298   0.718282       0.0649841

------------------------------------------------------------

=== Метод Рунге-Кутта 4-го порядка (p=4) ===
Достигнута точность ε = 1.00e-01 при эффективном шаге h = 0.1
Погрешность: 1.90717e-06
  x       y_num      y_true    погрешность
---  ----------  ----------  -------------
0    0           0             0
0.1  0.00517083  0.00517092    8.47423e-08
0.2  0.0214026   0.0214028     1.87309e-07
0.3  0.0498585   0.0498588     3.10513e-07
0.4  0.0918242   0.0918247     4.57561e-07
0.5  0.148721    0.148721      6.32103e-07
0.6  0.222118    0.222119      8.38299e-07
0.7  0.313752    0.313753      1.08087e-06
0.8  0.42554     0.425541      1.3652e-06
0.9  0.559601    0.559603      1.69738e-06
1    0.71828     0.718282      2.08432e-06

------------------------------------------------------------

=== Метод Милна ===
Достигнута точность ε = 1.00e-01 при эффективном шаге h = 0.2
Погрешность: 2.19443e-05
  x     y_num     y_true    погрешность
---  --------  ---------  -------------
0    0         0            0
0.2  0.0214    0.0214028    2.75816e-06
0.4  0.091818  0.0918247    6.73764e-06
0.6  0.222106  0.222119     1.2344e-05
0.8  0.425527  0.425541     1.43614e-05
1    0.71826   0.718282     2.19443e-05

------------------------------------------------------------

=== Работа завершена ===
```



Вывод

В ходе лабораторной работы я рассмотрел и реализовал на языке Python численные методы решения обыкновенных дифференциальных уравнений: метод Эйлера, усовершенствованный метод Эйлера и метод Милна.