Университет ИТМО
МФ КТиУ, Ф ПИиКТ

Лабораторная работа №6
Дисциплина «Вычислительная математика»

Численное решение обыкновенных дифференциальных уравнений

Выполнил
Аскаров Эмиль Рамилевич

Преподаватель:
Машина Екатерина
Алексеевна

г. Санкт-Петербург
2024 г.

Программная реализация задачи

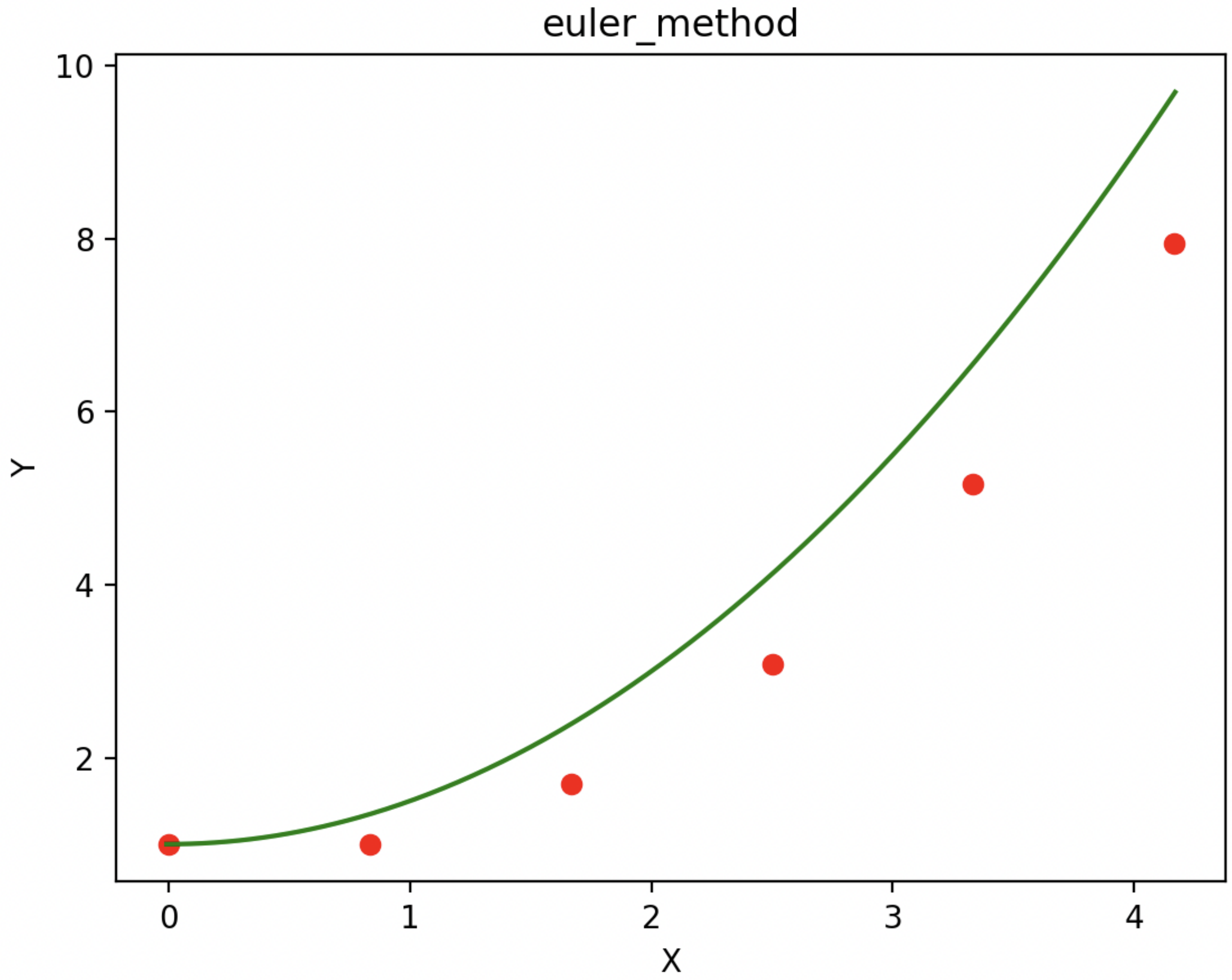
```
from matplotlib import pyplot as plt
from math import exp


def euler_method(f, xs, y0):
    ys = [y0]
    h = xs[1] - xs[0]
    for i in range(1, len(xs)):
        ys.append(ys[i - 1] + h * f(xs[i - 1], ys[i - 1]))
    return ys


def fourth_order_runge_kutta_method(f, xs, y0):
    ys = [y0]
    h = xs[1] - xs[0]
    for i in range(1, len(xs)):
        k1 = h * f(xs[i - 1], ys[i - 1])
        k2 = h * f(xs[i - 1] + h / 2, ys[i - 1] + k1 / 2)
        k3 = h * f(xs[i - 1] + h / 2, ys[i - 1] + k2 / 2)
        k4 = h * f(xs[i - 1] + h, ys[i - 1] + k3)
        ys.append(ys[i - 1] + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4))
    return ys


def adams_method(f, xs, y0):
    ys = fourth_order_runge_kutta_method(f, xs[:4], y0)
    h = xs[1] - xs[0]

    for i in range(4, len(xs)):
        df = f(xs[i - 1], ys[i - 1]) - f(xs[i - 2], ys[i - 2])
        d2f = f(xs[i - 1], ys[i - 1]) - 2 * f(xs[i - 2], ys[i - 2]) + f(xs[i - 3], ys[i -
3])
        d3f = f(xs[i - 1], ys[i - 1]) - 3 * f(xs[i - 2], ys[i - 2]) + 3 * f(xs[i - 3],
ys[i - 3]) - f(xs[i - 4], ys[i - 4])
        y = ys[i - 1] + h * f(xs[i - 1], ys[i - 1]) + h ** 2 / 2 * df + 5 * h ** 3 / 12 *
d2f + 3 * h ** 4 / 8 * d3f
        ys.append(y)
    return ys


def draw_plot(a, b, func, dx=0.01):
    xs, ys = [], []
    a -= dx
    b += dx
    x = a
    while x <= b:
        xs.append(x)
        ys.append(func(x))
        x += dx
    plt.plot(xs, ys, 'g')


def main(f, xs, y0, exact_y):
    methods = [euler_method,
               fourth_order_runge_kutta_method,
               adams_method]
    for method in methods:
        print(method.__name__)

        ys = method(f, xs, y0)
        if method in (adams_method,):
            inaccuracy = max([abs(exact_y(x) - y)
                              for x, y in zip(xs, ys)])
        else:
            xs2 = []
            for x1, x2 in zip(xs, xs[1:]):
                xs2.extend([x1, (x1 + x2) / 2, x2])
            ys2 = method(f, xs2, y0)
```

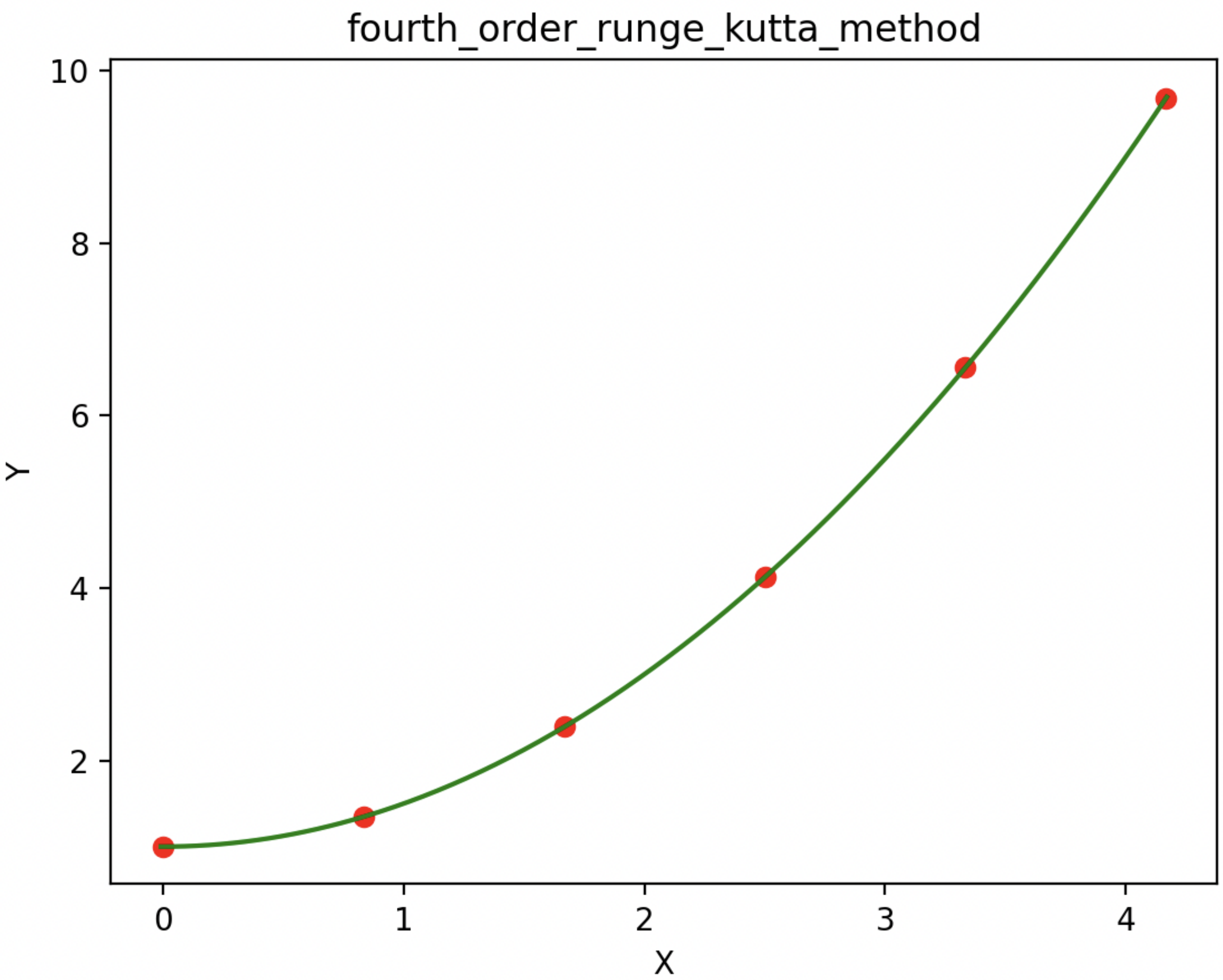
```python
            p = 4 if method is fourth_order_runge_kutta_method else 1
            inaccuracy = max([abs(y1 - y2) / (2 ** p - 1) for y1, y2 in zip(ys, ys2)])
        print("ys:", *map(lambda x: round(x, 5), ys))
        print(f"inaccuracy = {inaccuracy}")

        plt.title(method.__name__)

        draw_plot(xs[0], xs[-1], exact_y)
        for i in range(len(xs)):
            plt.scatter(xs[i], ys[i], c='r')
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.show()
        print('-' * 30)


def read_number(s: str):
    while True:
        try:
            return float(input(s))
        except Exception:
            continue


if __name__ == '__main__':
    print("1. y' = x")
    print("2. y' = e ** x")
    print("3. y' = x ** 2")
    mode = read_number("Выберите функцию: ")
    n = read_number("Введите n: ")
    x0 = read_number("Введите x0: ")
    xn = read_number("Введите xn: ")
    h = (xn - x0) / n
    xs = [x0 + h * i for i in range(int(n))]
    try:
        if mode == 1:
            # xs = [0, 1, 2, 3, 4, 5]
            f = lambda x, y: x
            y0 = 1
            exact_y = lambda x: x ** 2 / 2 + 1
        elif mode == 2:
            # xs = [0, 1, 2, 3, 4, 5]
            f = lambda x, y: exp(x)
            y0 = 0
            exact_y = lambda x: exp(x) - 1
        elif mode == 3:
            # xs = [0, 1, 2, 3, 4, 5]
            f = lambda x, y: x ** 2
            y0 = 5
            exact_y = lambda x: x ** 3 / 3 + 5
    except (ZeroDivisionError, ArithmeticError) as e:
        print("Функция не определена")

    main(f, xs, y0, exact_y)
```

Тестовые данные

```
1. y' = x
2. y' = e ** x
3. y' = x ** 2
Выберите функцию: 1
Введите n: 6
Введите x0: 0
Введите xn: 5
```

```
euler_method
ys: 1 1.0 1.69444 3.08333 5.16667 7.94444
inaccuracy = 5.555555555555557
------------------------------
fourth_order_runge_kutta_method
ys: 1 1.34722 2.38889 4.125 6.55556 9.68056
inaccuracy = 0.45717592592592593
------------------------------
adams_method
ys: 1 1.34722 2.38889 4.125 6.49769 9.56481
inaccuracy = 0.11574074074074225
------------------------------
```


euler_method
10
8
6
4
2
Y
0
1
2
3
4
X



Вывод

В ходе лабораторной работы я познакомился с численным решением обыкновенных дифференциальных уравнений.

Метод Эйлера – простой, но неточный метод, одношаговый.
Модификация метода Эйлера – более точный чем оригинал.
Методы Рунге-Кутта – хороший метод, но требует много вычислений по сравнению с прошлыми двумя, одношаговый.
Метод Адамса – многошаговый метод, точный. Использует на каждом шаге результаты предыдущих четырёх шагов. Использует конечные разности.
Метод Милна – многошаговый метод прогноза и коррекции. Коррекция проводится до тех пор, пока она не будет похожа на прогноз. Тоже использует результаты предыдущих четырёх шагов.