Национальный исследовательский университет ИТМО

Факультет ПиИКТ

Лабораторная работа № 6

по «Вычислительной математике»

Численное решение обыкновенных дифференциальных уравнений

Работу выполнил: Велюс Арина

Группа: Р32151

Преподаватель: Машина Екатерина Александровка

Санкт-Петербург

2023 г.

Цель работы:

решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Вариант №2

2. Усовершенствованный метод Эйлера,
3. Метод Рунге-Кутта 4- го порядка
5. Милна

Задание лабораторной работы:

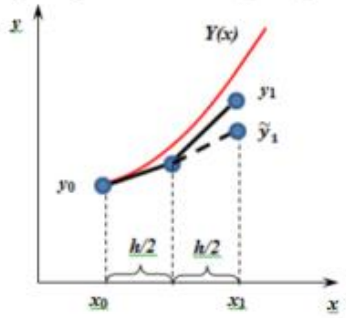
1. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;
2. Пользователь выбирает ОДУ вида $y' = f(x, y)$ (не менее трех уравнений), из тех, которые предлагает программа;
3. Предусмотреть ввод исходных данных 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;
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: $R = \frac{y^h - y^{h/2}}{2^p - 1} \leq \varepsilon$;
7. Для оценки точности многошаговых методов использовать точное решение задачи: $\varepsilon = \max_{0 \leq i \leq n} | y_{i\ точн} - y_i|$;
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.

Описание метода, расчетные формулы:

Формула модифицированного метода Эйлера:

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

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

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

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 $k = 4$

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

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

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

Листинг программы:

```
class Differentiator:
    """
    Класс, предоставляющий методы для численного
дифференцирования и решения обыкновенных дифференциальных
уравнений.
    """
    @staticmethod
    def modified_euler_method(x_start, x_end, y_start, h, f,
eps=None, log=False, digits=4):
        """
        Метод модифицированного метода Эйлера для численного
решения обыкновенного дифференциального уравнения.

        Параметры:
            x_start (float): Начальное значение x.
            x_end (float): Конечное значение x.
            y_start (float): Начальное значение y.
            h (float): Шаг.
            f (function): Функция, описывающая правую часть
дифференциального уравнения.
            eps (float, optional): Точность для метода Милна. По
умолчанию None.
            log (bool, optional): Флаг для вывода промежуточных
значений. По умолчанию False.
            digits (int, optional): Количество знаков после
запятой при выводе чисел. По умолчанию 4.

        Возвращает:
            x_list (list): Список значений x.
            y_list (list): Список значений y.
        """
        indexes = [0]
        x_list = Differentiator.__fill_x(x_start, x_end, h,
digits)
        y_list = [y_start]
        f_xy_list = [f(x_start, y_start)]
        exact_x, exact_y = Differentiator.get_exact_value(f,
x_start, x_end, y_start, h)
        exact_x, exact_y =
Differentiator.remove_from_exact(exact_x, exact_y, x_list)

        for i in range(1, len(x_list)):
            x, y = x_list[i - 1], y_list[i - 1]
            f_xy = f_xy_list[-1]

            next_x = x_list[i]
            next_y = y + h / 2 * (f_xy + f(next_x, y + h *
f_xy))
            next_f_xy = f(next_x, next_y)

            indexes.append(i)
            y_list.append(next_y)
```

```
            f_xy_list.append(next_f_xy)

        if log:
            pretty_table = PrettyTable()
            pretty_table.field_names = ["i", "xᵢ", "yᵢ", "f(xᵢ, 
yᵢ)", "Точное решение"]
            for i in range(len(x_list)):
                x = Differentiator.__to_fixed(x_list[i], digits)
                y = Differentiator.__to_fixed(y_list[i], digits)
                f_xy = Differentiator.__to_fixed(f_xy_list[i], 
digits)
                exact = Differentiator.__to_fixed(exact_y[i], 
digits)
                pretty_table.add_row([i, x, y, f_xy, exact])
            print(pretty_table)

        return x_list, y_list

    @staticmethod
    def runge_kutte_method(x_start, x_end, y_start, h, f, 
eps=None, log=False, digits=4):
        """
            Метод Рунге-Кутты для численного решения 
обыкновенного дифференциального уравнения.

            Параметры:
                x_start (float): Начальное значение x.
                x_end (float): Конечное значение x.
                y_start (float): Начальное значение y.
                h (float): Шаг.
                f (function): Функция, описывающая правую часть 
дифференциального уравнения.
                eps (float, optional): Точность для метода Милна. 
По умолчанию None.
                log (bool, optional): Флаг для вывода 
промежуточных значений. По умолчанию False.
                digits (int, optional): Количество знаков после 
запятой при выводе чисел. По умолчанию 4.

            Возвращает:
                x_list (list): Список значений x.
                y_list (list): Список значений y.
        """
        indexes = []
        x_list = Differentiator.__fill_x(x_start, x_end, h, 
digits)
        y_list = [y_start]
        f_xy_list = []
        exact_x, exact_y = Differentiator.get_exact_value(f, 
x_start, x_end, y_start, h)
        exact_x, exact_y = 
Differentiator.remove_from_exact(exact_x, exact_y, x_list)
        k_list = []
```

```
        for i in range(len(x_list)):
            x, y = x_list[i], y_list[i]

            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)

            next_y = y + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)

            indexes.append(i)
            y_list.append(next_y)
            f_xy_list.append(f(x, y))
            k_list.append([k1, k2, k3, k4])

        if log:
            pretty_table = PrettyTable()
            pretty_table.field_names = ["i", "xᵢ", "yᵢ", "k1",
"k2", "k3", "k4", "f(xᵢ, yᵢ)", "Точное решение"]
            for i in range(len(x_list)):
                k1, k2, k3, k4 = (Differentiator.__to_fixed(k,
digits) for k in k_list[i])
                x = Differentiator.__to_fixed(x_list[i], digits)
                y = Differentiator.__to_fixed(y_list[i], digits)
                f_xy = Differentiator.__to_fixed(f_xy_list[i],
digits)
                exact = Differentiator.__to_fixed(exact_y[i],
digits)
                pretty_table.add_row([i, x, y, k1, k2, k3, k4,
f_xy, exact])
            print(pretty_table)

        return x_list, y_list

    @staticmethod
    def milne_method(x_start, x_end, y_start, h, f, eps,
log=False, digits=4):
        """
              Метод Милна для численного решения обыкновенного
дифференциального уравнения.

              Параметры:
                  x_start (float): Начальное значение x.
                  x_end (float): Конечное значение x.
                  y_start (float): Начальное значение y.
                  h (float): Шаг.
                  f (function): Функция, описывающая правую
часть дифференциального уравнения.
                  eps (float): Точность.
                  log (bool, optional): Флаг для вывода
промежуточных значений. По умолчанию False.
                  digits (int, optional): Количество знаков
```

```
после запятой при выводе чисел. По умолчанию 4.

              Возвращает:
                  x_list (list): Список значений x.
                  y_list (list): Список значений y.
                  exact_y (list): Список значений точного
решения.
        """
        if round(x_start + 3 * h, 5) > x_end:
            print("Необходимо минимум 4 значения x")
            return [], []

        x_list = Differentiator.__fill_x(x_start, x_end, h,
digits)
        _, y_list = Differentiator.runge_kutte_method(x_start,
x_list[3], y_start, h, f, log=log)

        indexes = []
        f_xy_list = []
        exact_x, exact_y = Differentiator.get_exact_value(f,
x_start, x_end, y_start, h)
        exact_x, exact_y =
Differentiator.remove_from_exact(exact_x, exact_y, x_list)

        f_xy = [f(x_list[i], y_list[i]) for i in range(3)]

        for i in range(4, len(x_list)):
            x = x_list[i]
            y_predicted = y_list[-4] + 4 * h * (2 * f_xy[0] -
f_xy[1] + 2 * f_xy[2]) / 3
            new_f_xy = f(x, y_predicted)
            y_corrected = y_list[-2] + h * (f_xy[1] + 4 *
f_xy[2] + new_f_xy) / 3

            while abs(y_corrected - y_predicted) > eps:
                y_predicted = y_corrected
                new_f_xy = f(x, y_predicted)
                y_corrected = y_list[-2] + h * (f_xy[1] + 4 *
f_xy[2] + new_f_xy) / 3
            y_list.append(y_predicted)
            f_xy = f_xy[1:]
            f_xy.append(new_f_xy)
            f_xy_list.append(new_f_xy)
            indexes.append(i)

        if log:
            pretty_table = PrettyTable()
            pretty_table.field_names = ["i", "xᵢ", "yᵢ", "f(xᵢ,
yᵢ)", "Точное решение"]
            for i in range(4, len(x_list)):
                x = Differentiator.__to_fixed(x_list[i], digits)
                y = Differentiator.__to_fixed(y_list[i], digits)
                f_xy = Differentiator.__to_fixed(f_xy_list[i -
```

```
4], digits)
                exact = Differentiator.__to_fixed(exact_y[i],
digits)
                pretty_table.add_row([i, x, y, f_xy, exact])
            print(pretty_table)

        return x_list, y_list, exact_y
```

Примеры и результаты работы программы:

Ввод:

```
2*x*y
1
5
1
1
0.1
```

Вывод:

Модифицированный метод Эйлера:

i	x_i	y_i	$f(x_i, y_i)$	Точное решение
0	1.0000	1.0000	2.0000	1.0000
1	2.0000	8.0000	32.0000	70.4762
2	3.0000	144.0000	864.0000	7873.0051
3	4.0000	4608.0000	36864.0000	4206751.1804
4	5.0000	230400.0000	2304000.0000	26417235795.3512

Оценки точности по правилу Рунге:
[0.0, -0.31666666666666665, -27.05625, -3784.55390625, -975867.3837890625]

Метод Рунге-Кутте:

i	x_i	y_i	k1	k2	k3	k4	$f(x_i, y_i)$	Точное решение
0	1.0000	1.0000	2.0000	6.0000	12.0000	52.0000	2.0000	1.0000
1	2.0000	16.0000	64.0000	240.0000	680.0000	4176.0000	64.0000	70.4762
2	3.0000	1029.3333	6176.0000	28821.3333	108080.0000	872874.6667	6176.0000	7873.0051
3	4.0000	193171.5556	1545372.4444	8692720.0000	40855784.0000	410489555.5556	1545372.4444	4206751.1804
4	5.0000	85381827.5556	853818275.5556	5635200618.6667	31932803505.7778	384218223999.9999	853818275.5556	26417235795.3512

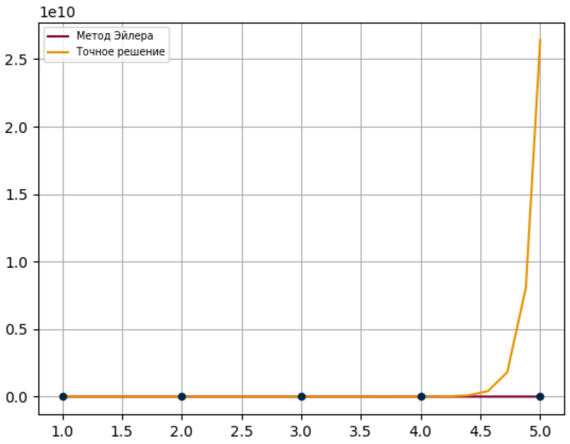
Оценки точности по правилу Рунге:
[0.0, -0.21315058955439808, -80.72882188872238, -72525.99159674464, -188747847.19167078]

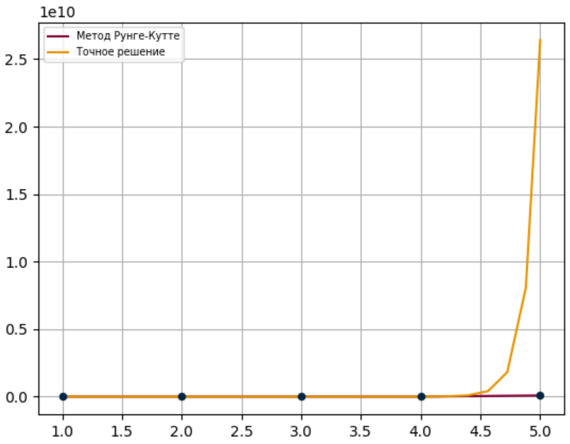
Метод Милна:

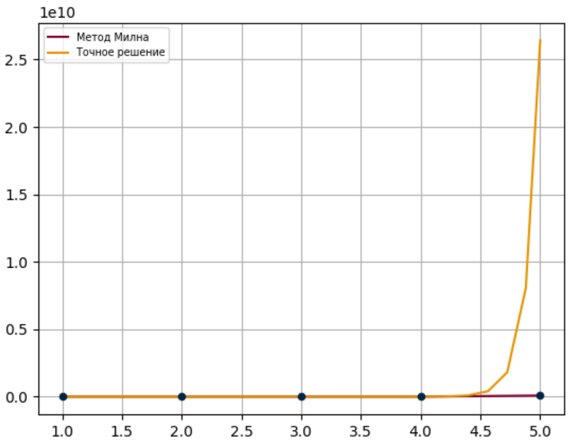
i	x_i	y_i	k1	k2	k3	k4	$f(x_i, y_i)$	Точное решение
0	1.0000	1.0000	2.0000	6.0000	12.0000	52.0000	2.0000	1.0000
1	2.0000	16.0000	64.0000	240.0000	680.0000	4176.0000	64.0000	70.4762
2	3.0000	1029.3333	6176.0000	28821.3333	108080.0000	872874.6667	6176.0000	7873.0051
3	4.0000	193171.5556	1545372.4444	8692720.0000	40855784.0000	410489555.5556	1545372.4444	3264218.2886

i	x_i	y_i	$f(x_i, y_i)$	Точное решение
4	5.0000	85381827.5556	inf	26417235795.3512

Точность: 26331853967.795624







Вывод:

В данной работе были рассмотрены методы решения задачи Коши для обыкновенных дифференциальных уравнений численными метода. Были реализованы следующие методы: модифицированный метод Эйлера, метод Рунге-Кутте, метод Милна.