Университет ИТМО
ФПИиКТ

Лабораторная работа №6
по Вычислительной математике

Выполнил: Балтабаев Дамир
Группа: P3210
Вариант: 3
Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург
2022

Цель лабораторной работы:

Решить задачу Коши численными методами.

Для исследования использовать:

- Одношаговые методы;
- Многошаговые методы.

Порядок выполнения:

3. Программная реализация задачи:

2.1. Исходные данные: ОДУ вида $y' = f(x, y)$, начальные условия $y(x_0)$, интервал дифференцирования $[a, b]$, шаг h, точность ε.

2.2. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям. Для оценки точности использовать правило Рунге.

2.3. Построить графики точного решения и полученного численного решения (разными цветами).

4. Анализ результатов работы: апробация и тестирование.

Рабочие формулы:

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

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

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

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

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

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:

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

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

2. по правилу Рунге:

$$R = \frac{y^h - y^{2h}}{2^p - 1},$$

где y^h - решение задачи Коши с шагом h в точке $x + 2h$

y^{2h} - решение задачи Коши с шагом $2h$ в точке $x + 2h$

p – порядок точности метода

Листинг программы

Одношаговый метод Рунге-Кутта 4-го порядка:

```
public Map<Double, Double> solve() {
    Map<Double, Double> y1 = new TreeMap<>();
    Map<Double, Double> y2 = new TreeMap<>();

    Double scale = Math.pow(10, 5);

    double x = a;
    y1.put(x, firstValue);
    for (int i = 1; i < n + 1; i++) {
        double k1 = h * function.calculate(x, y1.get(x));
        double k2 = h * function.calculate( x: x + h / 2,  y: y1.get(x) + k1 / 2);
        double k3 = h * function.calculate( x: x + h / 2,  y: y1.get(x) + k2 / 2);
        double k4 = h * function.calculate( x: x + h,  y: y1.get(x) + k3);
        y1.put(Math.round((x + h) * scale) / scale, (y1.get(x) + ((double) 1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)));
        x = Math.round((x + h) * scale) / scale;
    }

    while (true) {
        y2.clear();
        this.h = this.h / 2;
        this.n = calculateN(this.h);
        x = a;
        y2.put(x, firstValue);
        for (int i = 1; i < n + 1; i++) {
            double k1_r = h * function.calculate(x, y2.get(x));
            double k2_r = h * function.calculate( x: x + h / 2,  y: y2.get(x) + k1_r / 2);
            double k3_r = h * function.calculate( x: x + h / 2,  y: y2.get(x) + k2_r / 2);
            double k4_r = h * function.calculate( x: x + h,  y: y2.get(x) + k3_r);
            y2.put(Math.round((x + h) * scale) / scale, (y2.get(x) + ((double) 1 / 6) * (k1_r + 2 * k2_r + 2 * k3_r + k4_r)));
            x = Math.round((x + h) * scale) / scale;
        }
```

```
        int counter = 0;
        for (Double key : y1.keySet()) {
            if (Math.abs(y2.get(key) - y1.get(key)) / 15 < e) {
                counter++;
                this.rungeRuleValues.put(key, Math.abs(y2.get(key) - y1.get(key)));
            }
        }

        if (counter == y1.size()) {
            printTable(y1);
            graphics.drawFunction(y1);
            return y1;
        }
        this.rungeRuleValues.clear();
        y1.clear();
        y1 = y2;
    }

}
```

Многошаговый метод Милна:

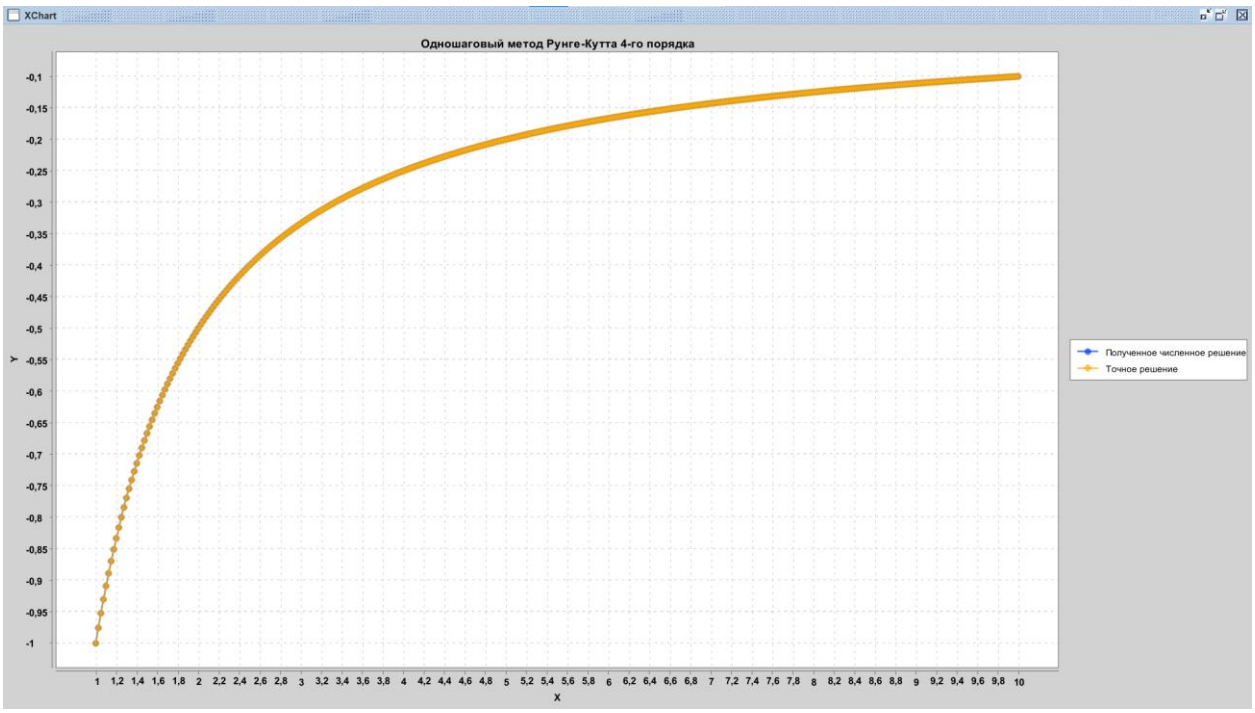
```
public void solve(Map<Double, Double> rungeKuttaMap, Map<Double, Double> rungeRuleMap) {
    Vector<Double> xRungeKutta = new Vector<>();
    Vector<Double> yRungeKutta = new Vector<>();
    Vector<Double> rungeRuleValues = new Vector<>();

    Double scale = Math.pow(10, 5);

    int i = 0;
    int coeff = 0;
    int counter = 0;
    for (Double key : rungeKuttaMap.keySet()) {
        if (counter == 4) break;

        if (Math.round((a + h * coeff) * scale) / scale == key) {
            xRungeKutta.add(i, key);
            yRungeKutta.add(i, rungeKuttaMap.get(key));
            rungeRuleValues.add(i, rungeRuleMap.get(key));
            i++;
            coeff++;
            counter++;
        }
    }


    Double x = xRungeKutta.lastElement();

    Double yProgn;
    Double yCorr;
```

```
    for (int j = 4; j < n + 1; j++) {
        x = Math.round((x + h) * scale) / scale;
        yProgn = yRungeKutta.get(j - 4) + ((4 * h) / 3) * (2 * function.calculate(xRungeKutta.get(j - 3), yRungeKutta.get(j - 3))
                - function.calculate(xRungeKutta.get(j - 2), yRungeKutta.get(j - 2))
                + 2 * function.calculate(xRungeKutta.get(j - 1), yRungeKutta.get(j - 1)));

        yCorr = yRungeKutta.get(j - 2) + (h / 3) * (function.calculate(xRungeKutta.get(j - 2), yRungeKutta.get(j - 2))
                + 4 * function.calculate(xRungeKutta.get(j - 1), yRungeKutta.get(j - 1)) + function.calculate(x, yProgn));

        while (Math.abs((yCorr - yProgn)) / 15 > e) {
            yProgn = yCorr;
            yCorr = yRungeKutta.get(j - 2) + (h / 3) * (function.calculate(xRungeKutta.get(j - 2), yRungeKutta.get(j - 2))
                    + 4 * function.calculate(xRungeKutta.get(j - 1), yRungeKutta.get(j - 1)) + function.calculate(x, yProgn));
        }
        xRungeKutta.add(x);
        yRungeKutta.add(yCorr);
        rungeRuleValues.add(Math.abs((yCorr - yProgn)) / 15);
    }

    printTable(xRungeKutta, yRungeKutta, rungeRuleValues);
    graphics.drawFunction(xRungeKutta, yRungeKutta);
}
```

Результаты выполнения программы:

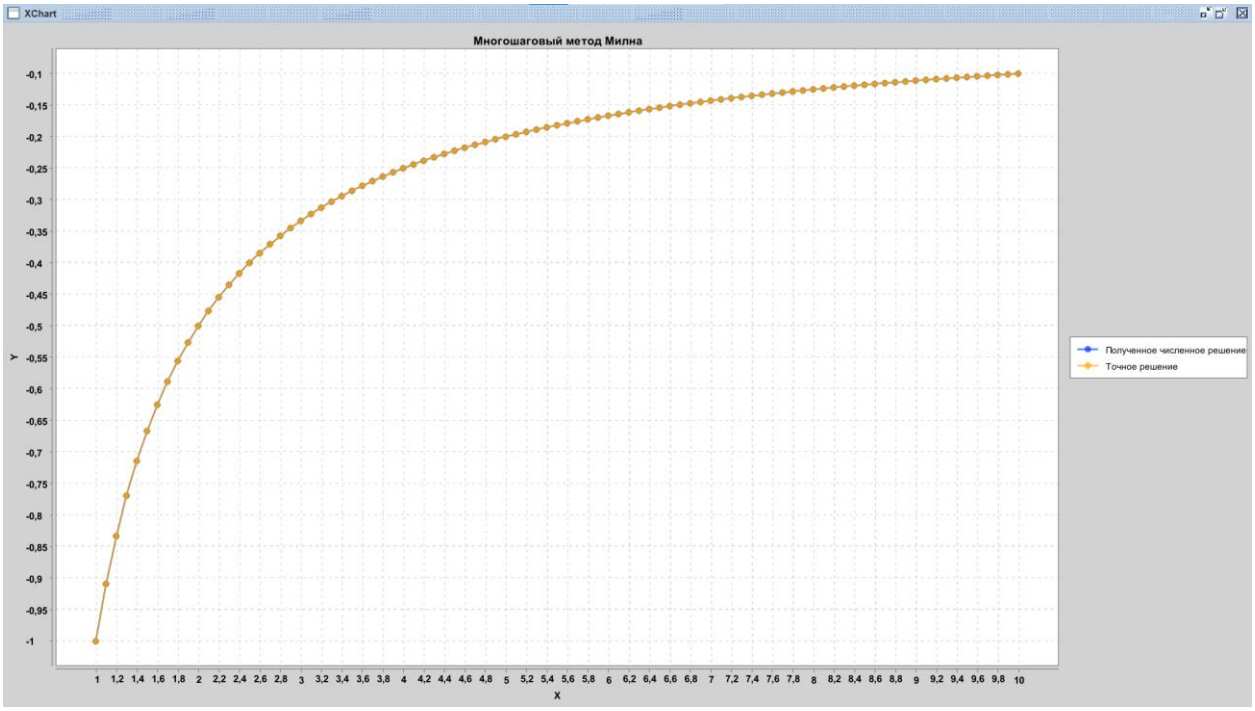
Одношаговый метод Рунге-Кутта четвертого порядка:

i	Xi	Yi	Оценка правилом Рунге
0	1.0	-1.0	0,000000000
1	1.025	-0.9756097588102673	0,000000000
2	1.05	-0.9523809573255356	0,000000000
3	1.075	-0.9302325649118065	0,000000000
4	1.1	-0.9090909173505819	0,000000000
5	1.125	-0.888888898349118	0,000000000
6	1.15	-0.8695652278100668	0,000000000
7	1.175	-0.8510638409601978	0,000000000
8	1.2	-0.8333333450879373	0,000000000
9	1.225	-0.8163265428027686	0,000000000
10	1.25	-0.8000000125034442	0,000000000
11	1.275	-0.7843137382028316	0,000000000
12	1.3	-0.7692307820652367	0,000000000
13	1.325	-0.7547169940149423	0,000000000
14	1.35	-0.7407407536104381	0,000000000
15	1.375	-0.727272740077804	0,000000000
16	1.4	-0.7142857269833552	0,000000000
17	1.425	-0.7017543985196797	0,000000000
18	1.45	-0.6896551847965542	0,000000000
19	1.475	-0.6779661138819275	0,000000000
20	1.5	-0.6666666786387963	0,000000000
21	1.525	-0.6557377166600857	0,000000000
22	1.55	-0.6451613018227296	0,000000000
23	1.575	-0.6349206461699273	0,000000000
24	1.6	-0.6250000109919354	0,000000000
25	1.625	-0.6153846261147823	0,000000000
26	1.65	-0.6060606165263691	0,000000000
27	1.675	-0.5970149355733648	0,000000000
28	1.7	-0.5882353040524909	0,000000000
29	1.725	-0.5797101545982138	0,000000000
30	1.75	-0.5714285808372047	0,000000000
31	1.775	-0.5633802908396044	0,000000000
32	1.8	-0.5555555644493481	0,000000000
33	1.825	-0.5479452141215854	0,000000000
34	1.85	-0.5405405489354439	0,000000000
35	1.875	-0.5333333441485771	0,000000000
36	1.9	-0.5263157973886846	0,000000000
37	1.925	-0.5194805271633137	0,000000000
38	1.95	-0.5128205202764823	0,000000000
39	1.975	-0.5063291211586682	0,000000000
40	2.0	-0.5000000070188466	0,000000000
41	2.025	-0.4938271673024758	0,000000000
42	2.05	-0.4878048846528239	0,000000000
43	2.075	-0.4819277172483859	0,000000000
44	2.1	-0.4761904824019933	0,000000000
45	2.125	-0.4705882413176	0,000000000
46	2.15	-0.4651162849106658	0,000000000
47	2.175	-0.4597701206061527	0,000000000
48	2.2	-0.4545454600370426	0,000000000



Многошаговый метод Милна:

i	Xi	Yi	Оценка правилом Рунге
0	1.0	-1.0	0,000000000
1	1.1	-0.9090909173505819	0,000000000
2	1.2	-0.8333333450879373	0,000000000
3	1.3	-0.7692307820652367	0,000000000
4	1.4	-0.7142830637011622	0,000000063
5	1.5	-0.6666657751018722	0,000000036
6	1.6	-0.6249968705907591	0,000000022
7	1.7	-0.5882346416709834	0,000000014
8	1.8	-0.5555525003148281	0,000000008
9	1.9	-0.5263157451357181	0,000000090
10	2.0	-0.4999971281809025	0,000000050
11	2.1	-0.4761910029171836 4	0,000000047
12	2.2	-0.4545426776126376	0,000000021
13	2.3	-0.4347836514607145 7	0,000000028
14	2.4	-0.4166638925293018 6	0,000000008
15	2.5	-0.400001512717 63	0,000000020
16	2.6	-0.3846125230140148 3	0,000000009
17	2.7	-0.3703723239518615 4	0,000000015
18	2.8	-0.357139897110410 3	0,000000060
19	2.9	-0.3448299529714804 5	0,000000013
20	3.0	-0.3333301580879659	0,000000006
21	3.1	-0.3225834234209584	0,000000012
22	3.2	-0.3124965569866968 7	0,000000008
23	3.3	-0.3030335006840252 4	0,000000012
24	3.4	-0.2941138737817235	0,000000009
25	3.5	-0.2857179186728039 4	0,000000012
26	3.6	-0.2777736546230769	0,000000010
27	3.7	-0.2702743612572239	0,000000012
28	3.8	-0.2631533619694307	0,000000011
29	3.9	-0.2564148341153611	0,000000013
30	4.0	-0.2499950110117119 5	0,000000012
31	4.1	-0.2439075375456295 5	0,000000013
32	4.2	-0.2380897448900958 8	0,000000013
33	4.3	-0.2325637980470976 8	0,000000014
34	4.4	-0.2272666797079409	0,000000014
35	4.5	-0.2222284848204819	0,000000015
36	4.6	-0.2173846494860508 8	0,000000015
37	4.7	-0.2127728731251451 7	0,000000016
38	4.8	-0.2083260148693283 6	0,000000017
39	4.9	-0.2040892553896014 3	0,000000017
40	5.0	-0.1999919577338953	0,000000018
41	5.1	-0.1960868203003 51	0,000000019
42	5.2	-0.1922988616439611 8	0,000000019
43	5.3	-0.1886884649321307	0,000000020
44	5.4	-0.1851754966414787 5	0,000000021
45	5.5	-0.1818283024190612	0,000000022
46	5.6	-0.1785608072850756 7	0,000000022
47	5.7	-0.1754496943330 7	0,000000023
48	5.8	-0.1724021583260114	0,000000024
49	5.9	-0.1695036832637660 5	0,000000025
50	6.0	-0.1666539123875766	0,000000026



Вывод:

В результате выполнения данной лабораторной работы, я познакомился с методами решения ОДУ. Узнал о понятиях “одношаговый метод” и “многошаговый метод”, поработал с одношаговым методом Рунге-Кутта 4-го порядка, а также с многошаговым методом Милна и начертил их графики, после чего сравнил их с графиком точного значения. Для оценки погрешностей использовал правило Рунге.