Университет ИТМО

Факультет ФПИ и КТ

Лабораторная работа №4

“Аппроксимация функции методом наименьших квадратов”

По вычислительной математике

Вариант 8

Выполнил: Рогачев М. С.

Группа: P32082

Преподаватель: Машина Е. А.

Санкт-Петербург

2023

1) Задание

1. Порядок выполнения работы

2. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;
3. Пользователь выбирает ОДУ вида $y' = f(x, y)$ (не менее трех уравнений), из тех, которые предлагает программа;
4. Предусмотреть ввод исходных данных с клавиатуры: начальные условия $y_0 = y(x_0)$, интервал дифференцирования $[x_0, x_n]$, шаг h, точность ε;
5. Для исследования использовать одношаговые методы и многошаговые методы (см. табл.1);
6. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям, для всех методов, реализуемых в программе;
7. Для оценки точности одношаговых методов использовать правило Рунге: $R = \frac{y^h - y^{h/2}}{2^p - 1} \le \varepsilon$;
8. Для оценки точности многошаговых методов использовать точное решение задачи: $\varepsilon = \max_{0 \le i \le n} |y_{i\,точн} - y_i|$;
9. Построить графики точного решения и полученного приближенного решения (разными цветами);
10. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных.
11. Проанализировать результаты работы программы.

2) Вычислительная реализация

Отсутствует ---

3) Листинг программы

Усовершенствованный метод Эйлера

```
import java.util.LinkedList;

public class ModifiedEulerMethod {
    public static double modifiedEulerMethod(double x0, double y0,
double h, double x, int n) {
        double y = y0;

        while (x0 < x) {
            double k1 = Function.getFunction(x0, y, n);
            double k2 = Function.getFunction(x0 + h, y + h * k1, n);

            y += h * (k1 + k2) / 2;
            x0 += h;
        }
```

```
        return y;
    }

    public static double solve(double x0, double y0, double h, double
xn, double eps, int num) {
        LinkedList<Double> x = new LinkedList<>();
        LinkedList<Double> y = new LinkedList<>();
        LinkedList<Double> y1 = new LinkedList<>();

        x.addLast(x0);
        y.addLast(y0);
        y1.addLast(y0);


        while (x.getLast() < xn) {
            double k1 = h * Function.getFunction(x.getLast(),
y1.getLast(), num);
            double k2 = h * Function.getFunction(x.getLast() + h,
y1.getLast() + k1 * h, num);

            y1.addLast(y1.getLast() + (k1 + k2) / 2);

            h /= 2;
            k1 = h * Function.getFunction(x.getLast(), y.getLast(),
num);
            k2 = h * Function.getFunction(x.getLast() + h, y.getLast() +
k1 * h, num);

            y.addLast(y.getLast() + (k1 + k2) / 2);

            h *= 2;
            x.addLast(x.getLast() + h);

            if (h > 1e-4 && Math.abs(y1.getLast() - y.getLast()) > eps)
{
                h /= 2;
                x.removeLast();
                y.removeLast();
                y1.removeLast();
            }
        }

        return y1.getLast();
    }
}
```

Метод Рунге-Кутта 4 порядка

```
import java.util.LinkedList;

public class RungeKuttaMethod {
    public static double rungeKuttaMethod(double x0, double y0, double
h, double xn, int n) {
        double y = y0;

        while (x0 < xn) {
```

```
            double k1 = h * Function.getFunction(x0, y, n);
            double k2 = h * Function.getFunction(x0 + h / 2, y + k1 / 2,
n);
            double k3 = h * Function.getFunction(x0 + h / 2, y + k2 / 2,
n);
            double k4 = h * Function.getFunction(x0 + h, y + k3, n);

            y += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
            x0 += h;
        }

        return y;
    }

    public static double solve(double x0, double y0, double h, double
xn, double eps, int num) {
        LinkedList<Double> x = new LinkedList<>();
        LinkedList<Double> y = new LinkedList<>();
        LinkedList<Double> y_runge = new LinkedList<>();

        x.addLast(x0);
        y.addLast(y0);
        y_runge.addLast(y0);


        while (x.getLast() < xn) {
            double k1 = h * Function.getFunction(x.getLast(),
y_runge.getLast(), num);
            double k2 = h * Function.getFunction(x.getLast() + h / 2,
y_runge.getLast() + k1 / 2, num);
            double k3 = h * Function.getFunction(x.getLast() + h / 2,
y_runge.getLast() + k2 / 2, num);
            double k4 = h * Function.getFunction(x.getLast() + h,
y_runge.getLast() + k3, num);

            y_runge.addLast(y_runge.getLast() + (k1 + 2 * k2 + 2 * k3 +
k4) / 6);

            h /= 2;
            k1 = h * Function.getFunction(x.getLast(), y.getLast(),
num);
            k2 = h * Function.getFunction(x.getLast() + h / 2,
y.getLast() + k1 / 2, num);
            k3 = h * Function.getFunction(x.getLast() + h / 2,
y.getLast() + k2 / 2, num);
            k4 = h * Function.getFunction(x.getLast() + h, y.getLast() +
k3, num);

            y.addLast(y.getLast() + (k1 + 2 * k2 + 2 * k3 + k4) / 6);

            h *= 2;
            x.addLast(x.getLast() + h);

            if (h > 1e-4 && Math.abs(y_runge.getLast() - y.getLast()) >
eps) {
                h /= 2;
```

```
                x.removeLast();
                y.removeLast();
                y_runge.removeLast();
            }
        }

        return y_runge.getLast();
    }
}
```

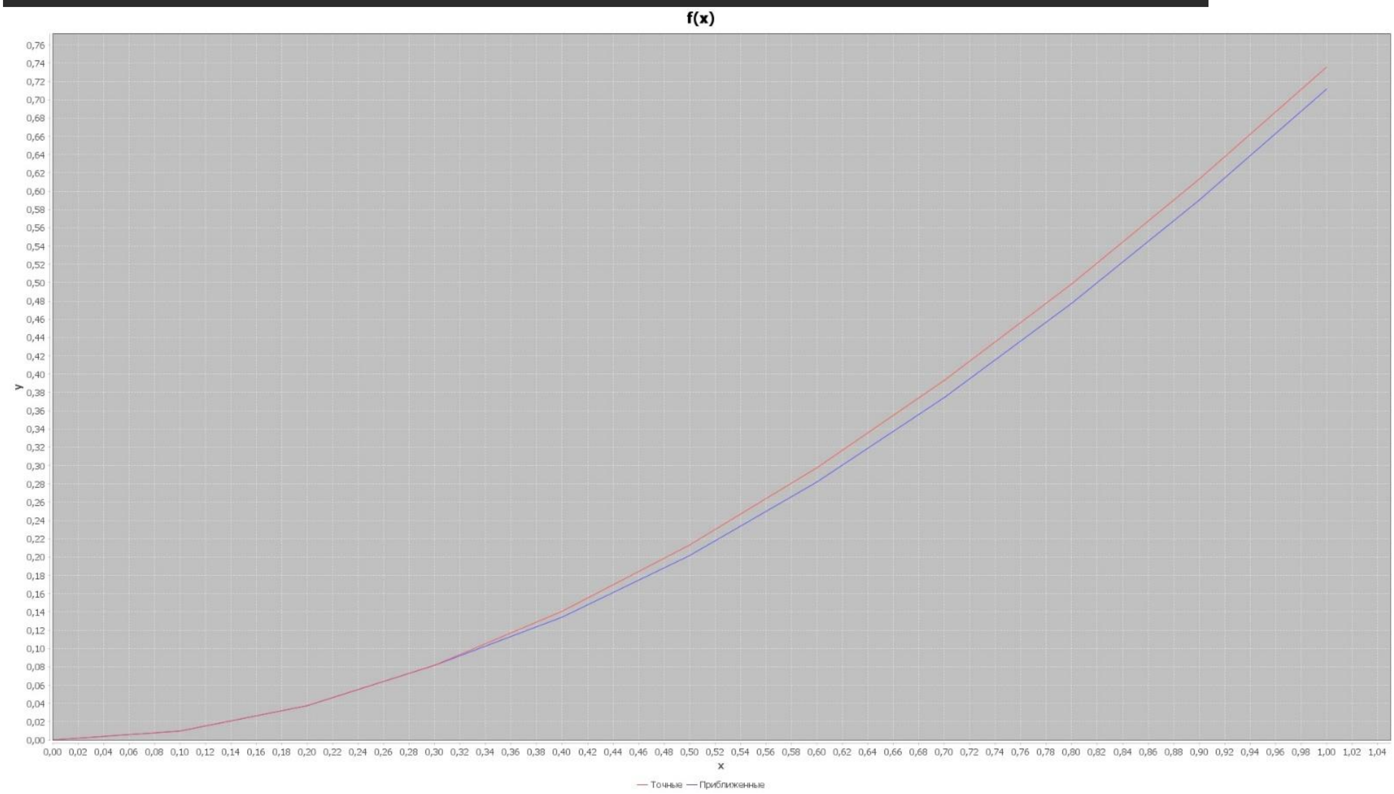
Метод Аддамса

```
import static java.lang.Math.pow;

public class AdamsMethod {
    public static double[] adamsMethod(double x0, double y0, double h,
double x, int num) {
        int n = (int) ((x - x0) / h);
        double[] y = new double[n + 1];
        double[] xArr = new double[n + 1];
        y[0] = y0;
        xArr[0] = x0;

        if (n < 4) {
            return null;
        }

        for (int i = 0; i < 4; i++) {
            y[i + 1] = RungeKuttaMethod.rungeKuttaMethod(xArr[i], y[i],
h, xArr[i] + h, num);
            xArr[i + 1] = xArr[i] + h;
        }

        for (int i = 4; i <= n; i++) {
            double k1 = Function.getFunction(xArr[i - 1], y[i - 1], num)
- Function.getFunction(xArr[i - 2], y[i - 2], num);
            double k2 = k1 + Function.getFunction(xArr[i - 3], y[i - 3],
num) - Function.getFunction(xArr[i - 2], y[i - 2], num);
            double k3 = k2 + 2 * Function.getFunction(xArr[i - 3], y[i -
3], num) - Function.getFunction(xArr[i - 2], y[i - 2], num) -
Function.getFunction(xArr[i - 4], y[i - 4], num);

            y[i] = y[i - 1] + h * Function.getFunction(xArr[i - 1], y[i
- 1], num) + (pow(h, 2) / 2) * k1 +
                    (5 * pow(h, 3) / 12) * k2 + (3 * pow(h, 4) / 8) *
k3;

            xArr[i] = xArr[i - 1] + h;
        }

        return y;
    }

    public static double[] solve(double x0, double y0, double h, double
x, double eps, int num) {
        int n = (int) ((x - x0) / h);
```

```
        double[] y = new double[n + 1];
        double[] xArr = new double[n + 1];
        y[0] = y0;
        xArr[0] = x0;

        if (n < 4) {
            return null;
        }

        for (int i = 0; i < 4; i++) {
            y[i + 1] = RungeKuttaMethod.solve(xArr[i], y[i], h, xArr[i]
+ h, eps, num);
            xArr[i + 1] = xArr[i] + h;
        }

        for (int i = 4; i <= n; i++) {
            y[i] = y[i - 1] + h * (55 * Function.getFunction(xArr[i -
1], y[i - 1], num) - 59 * Function.getFunction(xArr[i - 2], y[i - 2],
num) +
                    37 * Function.getFunction(xArr[i - 3], y[i - 3],
num) - 9 * Function.getFunction(xArr[i - 4], y[i - 4], num)) / 24;
            xArr[i] = xArr[i - 1] + h;

            double t = Double.MIN_VALUE;
            while (Math.abs(t - y[i]) > eps) {
                t = y[i];
                y[i] = y[i - 1] + h * (9 * Function.getFunction(xArr[i],
t, num) + 19 * Function.getFunction(xArr[i - 1], y[i - 1], num) -
                        5 * Function.getFunction(xArr[i - 2], y[i - 2],
num) + Function.getFunction(xArr[i - 3], y[i - 3], num)) / 24;
            }
        }

        return y;
    }
}
```

4) Примеры работы программы

```
Это шестая лабораторная работа по Вычислительной математике
Введите:
0 - для выхода
1 - для ввода данных из консоли
1
Введите номер функции:
1 - y' = 2x - y
2 - y' = y + (1 + x) * y
3 - y' = y * x + x * y
1
Введите x0:
0
Введите xn:
1
Введите y0:
0
Введите h:
0,1
Введите eps:
0,001
Модифифцированный Эйлер 0.8670591825487287
Модифифцированный Эйлер+ 0.735788350331101
Рунге 0.8657428307599382
Рунге+ 0.7357588824060002
Адамс 0.7116651444918161
Адамс+ 0.73578005106005
```



5) Вывод

В ходе работы я познакомился с различными методами численного решения дифференциальных уравнений, а также глубже узнал модифицированный метод Эйлера, метод Рунге-Кутты и метод Аддамса