УНИВЕРСИТЕТ ИТМО

Факультет Программной Инженерии и Компьютерных Технологий

Вычислительная математика

ЛАБОРАТОРНАЯ РАБОТА №5

Решение ОДУ

Усовершенствованный метод Эйлера

Преподаватель: Перл О.В.
Выполнил: Куприянов А.А, P3212

Санкт-Петербург
2020

Содержание

1 Теория **2**
1.1 Метод Эйлера 2
1.2 Усовершенствованный метод Эйлера 3

2 Реализация **5**
2.1 Блок-схема 5
2.2 Программа на Java 6

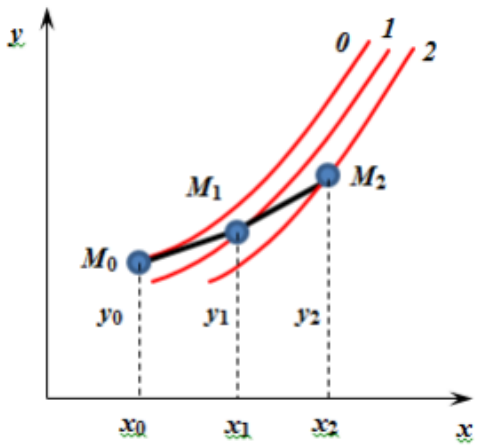
3 Результаты работы программы **7**
3.1 $y' = y$ 7
3.2 $y' = sin(x)$ 7
3.3 $y' = y(2sin(x) + 1)$ 7

4 Вывод **8**

1 Теория

Для того чтобы лучше понять усовершенствованный метод Эйлера, следует понять как работает его исходная версия

1.1 Метод Эйлера



Если уравнение

$$y^{'} = f(x, y),\ y(x_0) = y_0$$

не может быть решена аналитически, то нужно прибегнуть к численным методам для получения приближений к решению уравнения.

Мы можем вычислить приближенные значения в равноотстоящих точках $x_0, x_1, ..., x_n = b$ в интервале $[x_0, b]$, так что:

$$x_i = x_0 + ih,\ i = 0, 1, ..., n$$

где

$$h = \frac{b - x_0}{n}$$

Одним из методов позволяющих сделать это, является метод Эйлера

Тем не менее, это очень грубый метод, который редко используется на практике. Скорее, его используют для иллюистративных целей из-за его простоты.

Метод Эйлера основан на предположении, что касательная к интегральной кривой уравнения в $(x_i, y(x_i))$ апроксимирует интегральную кривую на отрезке $[x_i, x_{i+1}]$

Так как наклон интегральной кривой уравнения в точке $(x_i, y(x_i))$ равен $y(x_i)^{'} = f(x_i, y(x_i))$, уравнение касательной кривой в точке $(x_i, y(x_i))$ является:

$$y = y(x_i) + f(x_i, y(x_i))(x - x_i)$$

Пологая, что $x = x_{i+1} = x_i + h$, получим:

$$y_{i+1} = y(x_i) + hf(x_i, y(x_i))$$

как апроксимацию для значения $y(x_{i+1})$
И так как мы знаем начальное условие $y(x_0) = y_0$ мы можем использовать это уравнение с $i = 0$ для вычисления

$$y_1 = y_0 + hf(x0, y0)$$

Также и для $y2$, но следует заметить, что мы не можем вычислить $y2$ пока не вычислим $y1$.

Общий алгоритм Эйлера начинается с известного значения $y(x_O) = y_0$ и вычисления $y_1, y_2, ..., y_n$

$$y_{i+1} = y_i + hf(x_i, y_i),\ 0 \leq i \leq n-1$$

1.2 Усовершенствованный метод Эйлера

В методе Эйлера ошибка усечения равна $O(h)$. Можно предположить, что мы можем достичь высокой точности, просто выбрав достаточно маленький размер шага.

Но это не совсем так. Во-первых, чем меньше мы делаем шаг, тем больше ошибка округления (так как вычисления происходят на машинах, то присутствует машинный эпсилон, например). Во-вторых, вычисление $f(x, y)$ является довольно дорогим, поэтому мы не можем позволить себе слишком маленькие шаги.

Основным отличием усовершенствованного метода Эйлера является то, что при апроксимировании интегральной кривой в точке $(x_i, y(x_i))$ используется линия проходящая через $(x_i, y(x_i))$ с наклоном

$$m_i = \frac{f(x_i, y(x_i)) + f(x_{i+1}, y(x_{i+1}))}{2}$$

Тогда уравнение апроксимирующей линии превращается в

$$y = y(x_i) + \frac{f(x_i, y(x_i)) + f(x_{i+1}, y(x_{i+1}))}{2}(x - x_i)$$

Положив $x = x_{i+1} = x_i + h$ получим:

$$y_{i+1} = y(x_i) + \frac{h}{2}(f(x_i, y(x_i)) + f(x_{i+1}, y(x_{i+1})))$$

для апроксимации $y(x_{i+1})$

Мы уже знаем значение $y(x_i)$, но не значение $y(x_{i+1})$. Поэтому полученная схема является неявной. В этом случае, мы можем заменить $y(x_{i+1})$ на $y_i + hf(x_i, y_i)$ из метода Эйлера.

Тогда получим итоговую формулу:

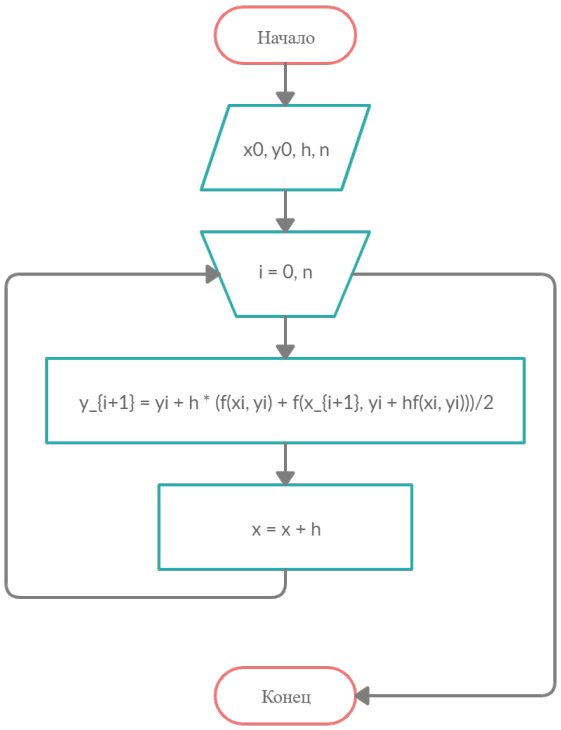
$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y(x_i)) + f(x_{i+1}, y_i + hf(x_i, y_i)))$$

Сравним апроксимацию к e этих двух методов:

n	метод Эйлера	Усовершенствованный метод Эйлера	Точное значение
12	2.613035290	2.707188994	2.718281828
24	2.663731258	2.715327371	2.718281828
48	2.690496599	2.717519565	2.718281828

2 Реализация

2.1 Блок-схема



2.2 Программа на Java

```
@Override
public List<Dot> solve(DiffEquation equation, double x0, double y0, double accuracy) {
        EulerSolverConfig config = calculateConfiguration(equation, x0, y0, accuracy);
        return solve(equation, x0, y0, config.pointsAmount, config.step);
    }

@Override
public List<Dot> solve(DiffEquation equation, double x0, double y0, int pointsAmount, double
    step) {
        List<Dot> dots = new ArrayList<>();
        dots.add(new SimpleDot(x0, y0));

        double x = x0;
        double calculatedY = y0;

        for (int i = 0; i < pointsAmount; i++) {
            double approximatedY = approximateNextY(equation, step, x, calculatedY);
            calculatedY = calculateY(equation, step, x, calculatedY, approximatedY);
            x += step;
            dots.add(new SimpleDot(x, calculatedY));
        }

        return dots;
}

private double approximateNextY(DiffEquation equation, double h, double x, double y){
        return y + h * equation.apply(x, y);
}

private double calculateY(DiffEquation equation, double h, double x, double y, double
    approximatedY){
        return y + h * (equation.apply(x, y) + equation.apply(x + h, approximatedY)) / 2;
}
```

3 Результаты работы программы

Эксперименты проводились на нескольких функциях

- $y' = y$
- $y' = sin(x)$
- $y' = y(2sin(x) + 1)$

3.1 $y' = y$

Общее решение уравнения имеет вид $y(x) = c_1 e^x$
Для задачи Коши используем несколько параметров и вычислим значение в $x = 0$

Начальные условия	Решение задачи	Точное значение	Интеролированное значение
$y(0.8) = -0.8$	$y(x) = -0.359463e^x$	-0.359463	-0.359493
$y(1) = 1$	$y(x) = e^{x-1}$	0,367879	0.367918
$y(0.1) = -0.1$	$y(x) = -0.0904837e^x$	-0.090483	-0.090484

3.2 $y' = sin(x)$

Общее решение уравнения имеет вид $y(x) = c_1 - cos(X)$
Для задачи Коши используем несколько параметров и вычислим значение в $x = 0$

Начальные условия	Решение задачи	Точное значение	Интеролированное значение
$y(0.8) = -0.8$	$y(x) = -cos(x) - 0.103293$	-1.103293	-1.103277
$y(1) = 1$	$y(x) = -cos(x) + 1 + cos(1)$	0.540302	0.540326
$y(0.1) = -0.1$	$y(x) = 0.895004 - cos(x)$	-0.104996	-0.104995

3.3 $y' = y(2sin(x) + 1)$

Общее решение уравнения имеет вид $y(x) = c_1 e^{(x-2cos(x))}$
Для задачи Коши используем несколько параметров и вычислим значение в $x = 0$

Начальные условия	Решение задачи	Точное значение	Интеролированное значение
$y(0.8) = -0.8$	$y(x) = -1.44813e^(x - 2cos(x))$	-0.195983	-0.196010
$y(1) = 1$	$y(x) = e^(x - 2cos(x) - 1 + 2cos(1))$	0.146695	0.146730
$y(0.1) = -0.1$	$y(x) = -0.661942e^(x - 2cos(x))$	-0.089584	-0.089606

4 Вывод

Численные методы могут оказать помощь в тех случаях, когда не работает аналитический подход. Тогда мы можем применить один из численных методов для приближенного решения.

Тем не менее, возникает два вида погрешностей в виду особенности среды исполнения и самих методов. Ошибка усечения является погрешностью самого метода, ведь, например, в методах Эйлера для решения ОДУ мы апроксимируем некоторые значения из-за чего возникает погрешность. Также есть ошибка округления связанная с машинными операциями.

Основываясь на них мы можем проводить оценку численных методов и выбрать наиболее подходящий из них.

Например, сравнивая метод Эйлера и усовершенствованный метод Эйлера мы можем выяснить, что несмотря на то, что мы можем получить сколь угодно хорошую точность уменьшая шаг - мы упираемся в характеристики вычислительной машины.

Усовершенствованный метод Эйлера при сравнении с другими одношаговыми методами оказался точнее, чем обычный метод Эйлера, но менее точным, чем метод Рунге-Кутта (метод Рунге-Кутты 4 порядка), хотя в методе Эйлера выполняется меньше операций.

Также сравнивая с многошаговыми методами, то с помощью метода Эйлера мы сразу можем начать счет по одному лишь известному значению y_0. Кроме того, например,метод Адамса не позволяет (без усложнения формул) изменить шаг в процессе счета. Такого недостатка нету в одношаговых методах