Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5
ОДУ и задача Коши

по курсу «Вычислительной математики»

Выполнил:
Студент группы P3230
Пономаренко Алиса Валерьевна

Преподаватель:
Перл Ольга Вячеславовна

Санкт-Петербург
2024

Описание численного метода

Метод Милна

Метод Милна относится к многошаговым методам и представляет один из методов прогноза и коррекции. Решение в следующей точке находится в два этапа. На первом этапе осуществляется по специальной формуле прогноз значения функции, а затем на втором этапе - коррекция полученного значения.
Если полученное значение y после коррекции существенно отличается от спрогнозированного, то проводят еще один этап коррекции. Если опять имеет место существенное отличие от предыдущего значения (т.е. от предыдущей коррекции), то проводят еще одну коррекцию и т.д. Однако очень часто ограничиваются одним этапом коррекции.

Описание метода Милна

Рассмотрим метод на примере уравнения:

$y' = f(x, y)$, $x \in [a, b]$

$y(a) = y_0$

Будем считать, что каким-либо одношаговым методом получен начальный отрезок решения-значения y_0, y_1, y_2, y_3.

Тогда известны y'_0, y'_1, y'_2, y'_3.

Первая формула Милна:

$$y_m = y_{m-4} + \frac{4h}{3}\left(2y'_{m-3} - y'_{m-2} + 2y'_{m-1}\right), \quad m = 4, 5, \ldots, N. \tag{1}$$

Здесь $\quad y'_{m-i} = f(x_{m-i}, y_{m-i}), \quad i = 1, 2, 3.$

Вторая формула Милна:

$$y_m = y_{m-2} + \frac{h}{3}\left(y'_{m-2} + 4y'_{m-1} + y'_m\right), \quad m = 2, 3, \ldots, N. \tag{2}$$

Первой формулой (1) мы предсказываем значение y_m, второй (2) - уточнаяем.

Алгоритм метода

Он состоит в следующем:

1. Зная y_0, по какой-либо формуле для одношаговых методов находим y_1, y_2, y_3. Это может быть метод Рунге-Кутты третьего или четвертого порядка точности;

2. Вычисляем $y_m^{[1]}$, $m = 4, 5, \ldots$, по первой формуле Милна

$$y_m^{[1]} = y_{m-4} + \frac{4h}{3}\left(2f_{m-3} - f_{m-2} + 2f_{m-1}\right), \quad m = 4, 5, \ldots, N,$$

$$f_i = f\left(x_i, y_i\right), \quad i = m-3, m-2, m-1;$$

3. Найденное значение $y_m^{[1]}$ подставляем в исходное уравнение и находим

$$y_m^{[1]\prime} = f\left(x_m, y_m^{[1]}\right) \equiv f'_m;$$

4. По второй формуле Милна получаем $y_m^{[2]}$:

$$y_m^{[2]} = y_{m-2} + \frac{h}{3}\left(f_{m-2} + 4f_{m-1} + f'_m\right), \quad m = 4, 5, \ldots, N;$$

5. Вычисляем

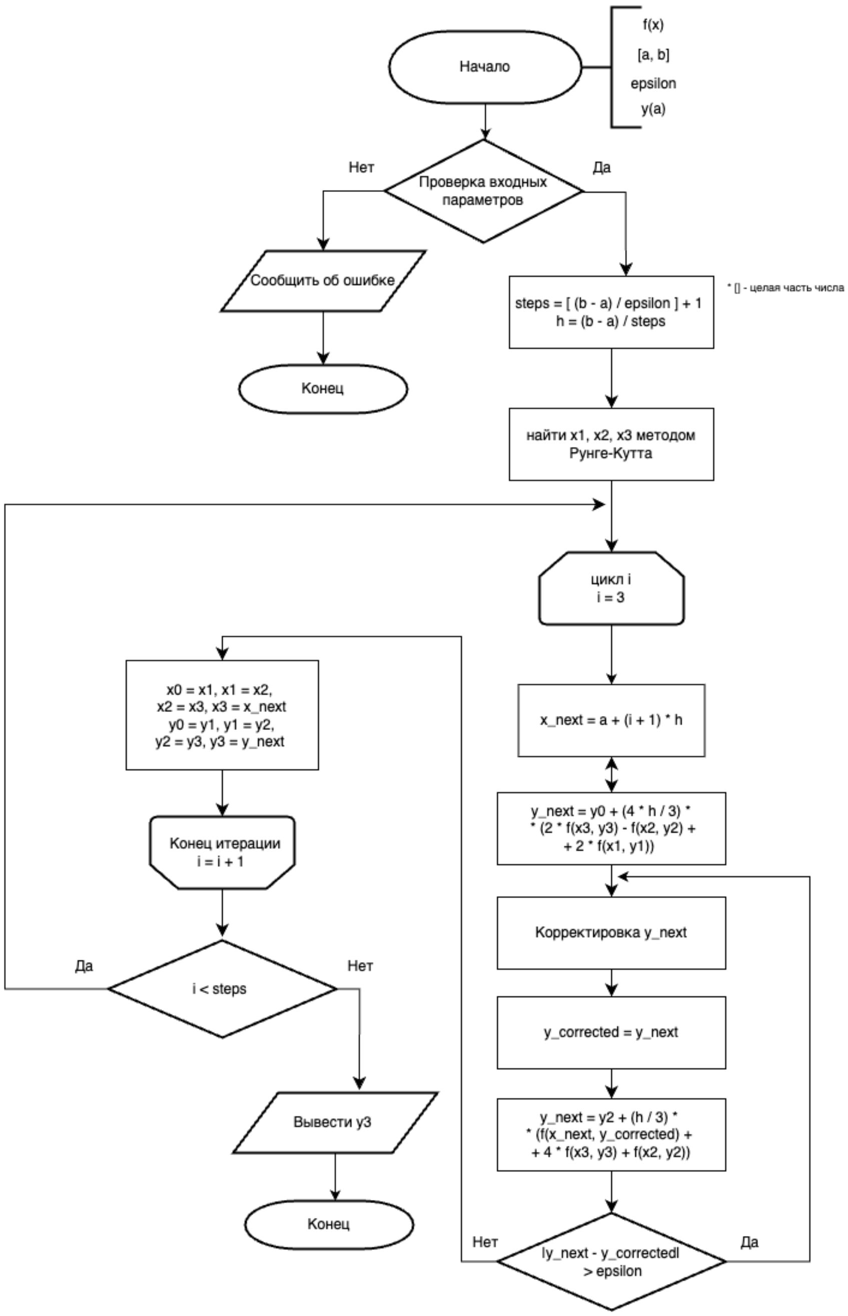
$$\epsilon_m = \left|y_m^{[2]} - y_m^{[1]}\right|.$$

Если $\epsilon_m \leq \epsilon$, где ϵ — принятая точность вычислений, то полагаем

$$y_m \approx y_m^{[2]}, \quad y'_m \approx f\left(x_m, y_m^{[2]}\right)$$

и переходим к следующей точке сетки. Если $\epsilon_m > \epsilon$ и точность не достигнута, следует уменьшить шаг h. При этом потребуется пересчитать начальные значения y_1, y_2, y_3.

Блок-схема по составленному описанию метода



Код численного метода

```
1 usage   new *
public static double solveByMilne(int f, double epsilon, double a, double y_a, double b) {
    if (epsilon <= 0 || a >= b) {
        throw new IllegalArgumentException("Invalid input parameters.");
    }
    BiFunction<Double, Double, Double> func = get_function(f);

    // Calculate the number of steps and step size h
    int steps = (int) Math.ceil((b - a) / epsilon) + 1;
    double h = (b - a) / steps;

    double[] x = new double[4];
    double[] y = new double[4];

    // Calculate the first three points using Runge-Kutta method
    x[0] = a;
    y[0] = y_a;
    for (int i = 1; i < 4; i++) {
        x[i] = x[0] + i * h;
        y[i] = getRungeKuttaFunctionValue(func, x[i - 1], y[i - 1], h);
    }

    return getYFinalValue(steps, func, h, y, x, epsilon, a);
}

1 usage   new *
private static double getYFinalValue(int steps, BiFunction<Double, Double, Double> func,
                                     double h, double[] y, double[] x, double epsilon, double a) {
    double y_next;
    for (int i = 3; i < steps; i++) {
        double x_next = a + (i + 1) * h;
        y_next = y[0] + (4 * h / 3) * (2 * func.apply(x[3], y[3]) - func.apply(x[2], y[2])
                + 2 * func.apply(x[1], y[1]));

        // Correct the predicted y_next using iterative method
        y_next = correctYNext(func, h, x, y, epsilon, x_next, y_next);

        // Shift arrays for the next iteration
        System.arraycopy(x, srcPos: 1, x, destPos: 0, length: 3);
        System.arraycopy(y, srcPos: 1, y, destPos: 0, length: 3);

        x[3] = x_next;
        y[3] = y_next;
    }
    return y[3];
}
```

```java
    1 usage   new *
    private static double correctYNext(BiFunction<Double, Double, Double> func, double h, double[] x,
                                       double[] y, double epsilon, double x_next, double y_next) {
        double y_corrected;
        do {
            y_corrected = y_next;
            y_next = y[2] + (h / 3) *
                    (func.apply(x_next, y_corrected) + 4 * func.apply(x[3], y[3]) + func.apply(x[2], y[2]));
        } while (Math.abs(y_next - y_corrected) > epsilon);
        return y_next;
    }

    /**
     * @return next y value
     */
    1 usage   new *
    private static double getRungeKuttaFunctionValue(BiFunction<Double, Double, Double> func, double x,
                                                     double y, double h) {
        double point1 = h * func.apply(x, y);
        double point2 = h * func.apply( t: x + h / 2,  u: y + point1 / 2);
        double point3 = h * func.apply( t: x + h / 2,  u: y + point2 / 2);
        double point4 = h * func.apply( t: x + h,  u: y + point3);
        return y + (point1 + 2 * point2 + 2 * point3 + point4) / 6;
    }

    new *
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(System.out));

        int f = Integer.parseInt(bufferedReader.readLine().trim());
        double epsilon = Double.parseDouble(bufferedReader.readLine().trim());
        double a = Double.parseDouble(bufferedReader.readLine().trim());
        double y_a = Double.parseDouble(bufferedReader.readLine().trim());
        double b = Double.parseDouble(bufferedReader.readLine().trim());

        double result = Result.solveByMilne(f, epsilon, a, y_a, b);

        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Примеры работы программы

Номер теста	Входные данные $[f, \epsilon, a, y(a), b]$	Выходные данные
1	2 0.1 0 1 1	1.2840257
2	2 0.0001 0 1 1	1.2840254
3	2 0.1 0 1.5 1	1.9260386
4	2 0.01 -1 -1 1	-1.0000000000001923
5	1 0.001 1 0 5	0.2566401
6	1 0.001 1 0 2	0.9564491
7	1 0.001 0 0 3.141592652	2.0000000000000124

1. В первых двух тестах разное значение ϵ, однако выходные данные отличаюся, начиная с 7 знака после запятой.

2. Четвертый тест - метод корректно отрабатывает с отрицательными входными данными - началом отрезка и значением функции в этой точке.

3. Перавая функция особая - в ней производная y не зависит от y ($y' = sin(x)$).

4. Пятый и шестой тесты - разные значения конца отрезка.

5. Пятая функция - с точками, которые выходят за область определения функции

6. Седьмой тест - значения 0 и π особые для тригонометрических функций.

Выводы

Метод правильно работает с особыми точками (для определенных функций), потому что использует предварительное значение для оценки следующего значения, а затем корректирует его на основе более точной аппроксимации. Это помогает улучшить точность и устойчивость метода, особенно в точках, где могут возникать ошибки.

Преимущества метода Милна

1. **Высокая точность**: Метод Милна обладает высокой точностью благодаря использованию корректирующего этапа, который снижает ошибки предиктора.
2. **Стабильность**: Предиктор-корректорный подход повышает устойчивость метода, особенно для гладких и периодических функций.
3. **Эффективность**: Метод эффективен для уравнений, где требуется высокая точность на больших интервалах времени.

Ограничения метода Милна

1. **Необходимость начальных значений**: Требуются несколько начальных значений, что может усложнять его применение на практике.
2. **Сложность реализации**: Метод сложнее в реализации по сравнению с простыми методами, такими как метод Эйлера.
3. **Чувствительность к выбору шага** h: Неправильный выбор шага может привести к накоплению ошибок и снижению точности.

Начальные условия: Для начала метода Милна необходимы несколько начальных значений, которые могут быть вычислены с помощью метода Рунге-Кутты или другого точного метода.

Сложность и вычислительная стоимость: Метод требует вычисления нескольких значений производных на каждом шаге, что может увеличивать вычислительную стоимость по сравнению с более простыми методами.

Сравнение с другими методами интегрирования

- **Метод Эйлера**

  Преимущества: Простой в реализации, требует минимальных вычислений.
  Недостатки: Низкая точность, особенно на больших интервалах времени; ошибки накапливаются быстрее.

- **Метод Рунге-Кутты (четвертого порядка)**

  Преимущества: Высокая точность, не требует начальных значений, легко применять.
  Недостатки: Большая вычислительная стоимость по сравнению с методом Эйлера.

- **Метод Адамса-Бэшфорта**

  Преимущества: Хорошо подходит для жестких уравнений, высокая точность.
  Недостатки: Требует начальных значений, сложнее в реализации, чем метод Эйлера и Рунге-Кутты.

- **Метод Милна**

  Преимущества: Высокая точность благодаря предиктору-корректору, устойчивая работа с гладкими и периодическими функциями.
  Недостатки: Требует нескольких начальных значений, сложнее в реализации, чувствительность к выбору шага.

Алгоритмическая сложность метода: $O(n)$, **где** $n = |b - a|/\epsilon$

Анализ численных ошибок метода Милна

Метод Милна обладает ошибкой порядка $O(h^4)$, что делает его эффективным для решения обыкновенных дифференциальных уравнений. Однако, стабильность метода зависит от выбора шага h и характеристик решаемого уравнения. Важно правильно выбирать шаг и начальные значения для достижения наилучших результатов.

Заключение

Метод Милна — это инструмент для численного решения ОДУ, обладающий высокой точностью и стабильностью. Он особенно полезен для решения гладких и периодических функций. Однако, его сложность и необходимость начальных значений могут ограничивать его применение. В сравнении с другими методами, такими как метод Эйлера и метод Рунге-Кутты, метод Милна предлагает высокую точность, но требует большего количества начальных значений и сложен в реализации. В целом, метод Милна является эффективным и точным выбором для задач, требующих высокого уровня точности на больших интервалах времени.