ИТМО Кафедра Информатики и прикладной математики

Отчет по лабораторной работе №4 «Решение ОДУ(задача Коши)» Вариант : метод предиктора и корректора

Выполнил: студент группы P3217
Плюхин Дмитрий
Преподаватель: Калёнова О. В.

2016 год

1. Описание метода

Схема предиктор-корректор (метод прогноза и коррекции, предсказывающе-исправляющий метод) — в вычислительной математике — семейство алгоритмов численного решения различных задач, которые состоят из двух шагов. На первом шаге (предиктор) вычисляется грубое приближение требуемой величины. На втором шаге при помощи иного метода приближение уточняется (корректируется).

Метод предиктора-корректора – многошаговый метод, это значит, что при расчете координат очередной точки учитываются несколько предыдущих точек, а не только одна, в отличие от одношаговых методов, таких как метод Эйлера и семейство методов Рунге-Кутты.

Метод предиктора-корректора, как и многие из многошаговых методов, основан на идее преобразования ОДУ следующим образом:

Предполагается, что изначально ОДУ имеет вид

$$\frac{dy}{dx} = f(x, y)$$

Дифференциал аргумента переносят вправо и берут интеграл от обеих частей.

$$\int_{y_i}^{y_{i+1}} dy = \int_{x_i}^{x_{i+1}} f(x, y)\, dx$$

Значение левой части получившегося выражения находится достаточно просто. Перенеся y_i в правую часть, получаем выражение для вычисления ординаты очередной точки

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y)\, dx$$

Остается посчитать интеграл от f(x,y). Для упрощения вычислений функцию f(x,y) заменяют каким-либо интерполяционным полиномом и, интегрируя, выводят общую формулу, в частности, для метода Адамса, где используется полином Ньютона, она имеет вид

$$y_{i+1} = y_i + \frac{h}{24}\left(59f(x_i, y_i) - 55f(x_{i-1}, y_{i-1}) + 37f(x_{i-2}, y_{i-2}) - 9f(x_{i-3}, y_{i-3})\right) \quad (1)$$

Где h – шаг сетки.

Однако, получаемое на этом этапе значение y является конечным результатом для метода Адамса, но не для метода предиктора и корректора. В нашем случае на данном этапе заканчивается только первая фаза («предиктор»), и для того, чтобы получить окончательное значение y, необходимо использовать еще одну формулу

$$y_{i+1} = y_i + \frac{h}{24}\left(19f(x_{i+1}, \widetilde{y_{i+1}}) + 9f(x_i, y_i) - 5f(x_{i-1}, y_{i-1}) + f(x_{i-2}, y_{i-2})\right) \quad (2)$$

Где $\widetilde{y_{i+1}}$ – значение, полученное на фазе «предиктор».

Фаза «корректор», как правило, позволяет повысить точность решения задачи Коши.

Как можно было заметить, применение метода предиктора и корректора, как и многих других многошаговых методов, нельзя начинать сразу же – предварительно необходимо найти каким-либо одношаговым методом минимально возможное количество точек (в данном случае 3, если не считать точку, задаваемую задачей Коши). Для этой цели можно использовать метод Рунге-Кутты 4-го порядка.

2. Листинг основной части программы

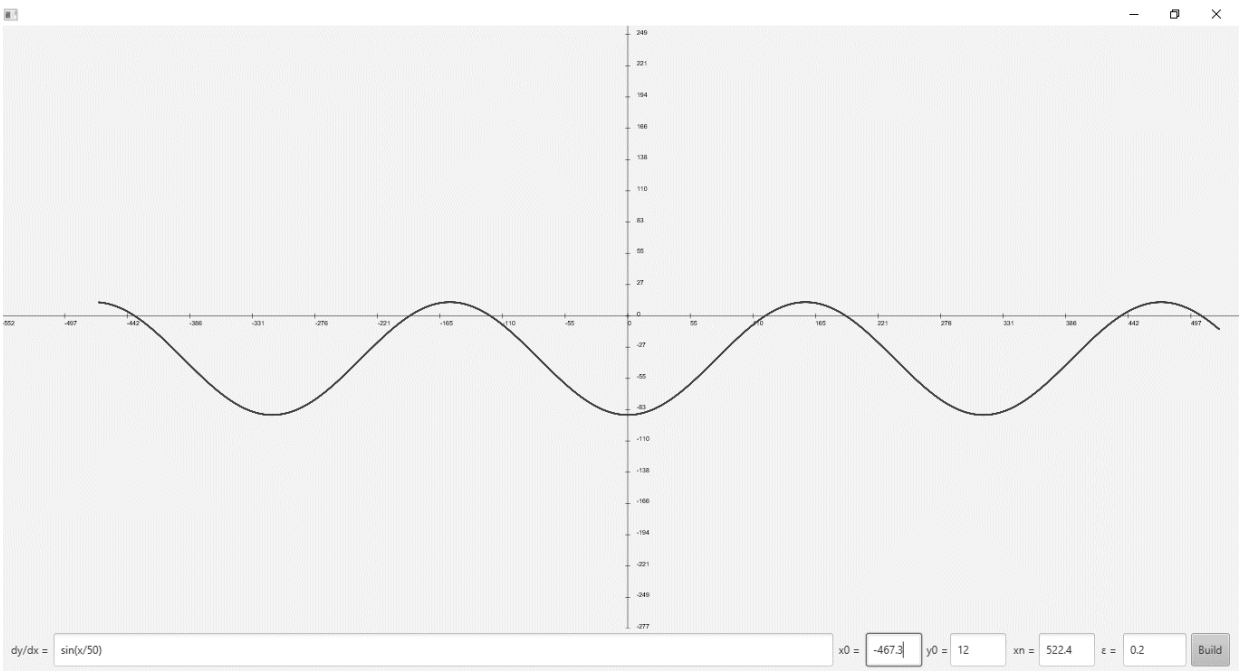
```
package sample;

import Calculator.Calculator;

import java.util.ArrayList;

/**
 * Created by Zerbs on 21.11.2016.
 */
public class MethodPredictoraCorrectora {
    private static final int beginNumOfPoints = 4;

    public static ArrayList<Point> findPoints(String expression, double x0, double y0, double
epsilon, double xn){
        ArrayList<Point> points = MethodRungeKutta.findPoints(expression, x0, y0, epsilon,
beginNumOfPoints);
        double h = MethodRungeKutta.getStep();

        ArrayList<Double> funcs = fillFirstFourFuncs(expression, points);
```

```
        String newexpression;
        double x = points.get(beginNumOfPoints-1).getX();
        double y;

        double func;

        int i = beginNumOfPoints;
        double y_pre;
        while (x < xn){
            x+=h;
            y_pre = points.get(i-1).getY() + h/24*(59*funcs.get(i-1)-55*funcs.get(i-
2)+37*funcs.get(i-3)-9*funcs.get(i-4));

            newexpression = expression.replace("x", x + "");
            newexpression = newexpression.replace("y", y_pre + "");
            func = Double.parseDouble(Calculator.Calculate(newexpression));

            y = points.get(i-1).getY() + h/24*(9*func + 19*funcs.get(i-1) - 5*funcs.get(i-2) +
funcs.get(i-3));
            newexpression = expression.replace("x", x + "");
            newexpression = newexpression.replace("y", y + "");
            func = Double.parseDouble(Calculator.Calculate(newexpression));

            funcs.add(func);
            points.add(new Point(x,y,0));
            i++;
        }
        return points;
    }

    private static ArrayList<Double> fillFirstFourFuncs(String expression, ArrayList<Point>
points){
        ArrayList<Double> funcs = new ArrayList<>();
        String newexpression;
        double func;
        for (int i = 0; i < 4; i++){
            newexpression = expression.replace("x", points.get(i).getX() + "");
            newexpression = newexpression.replace("y", points.get(i).getY() + "");
            func = Double.parseDouble(Calculator.Calculate(newexpression));
            funcs.add(func);
        }
        return funcs;
    }
}
```

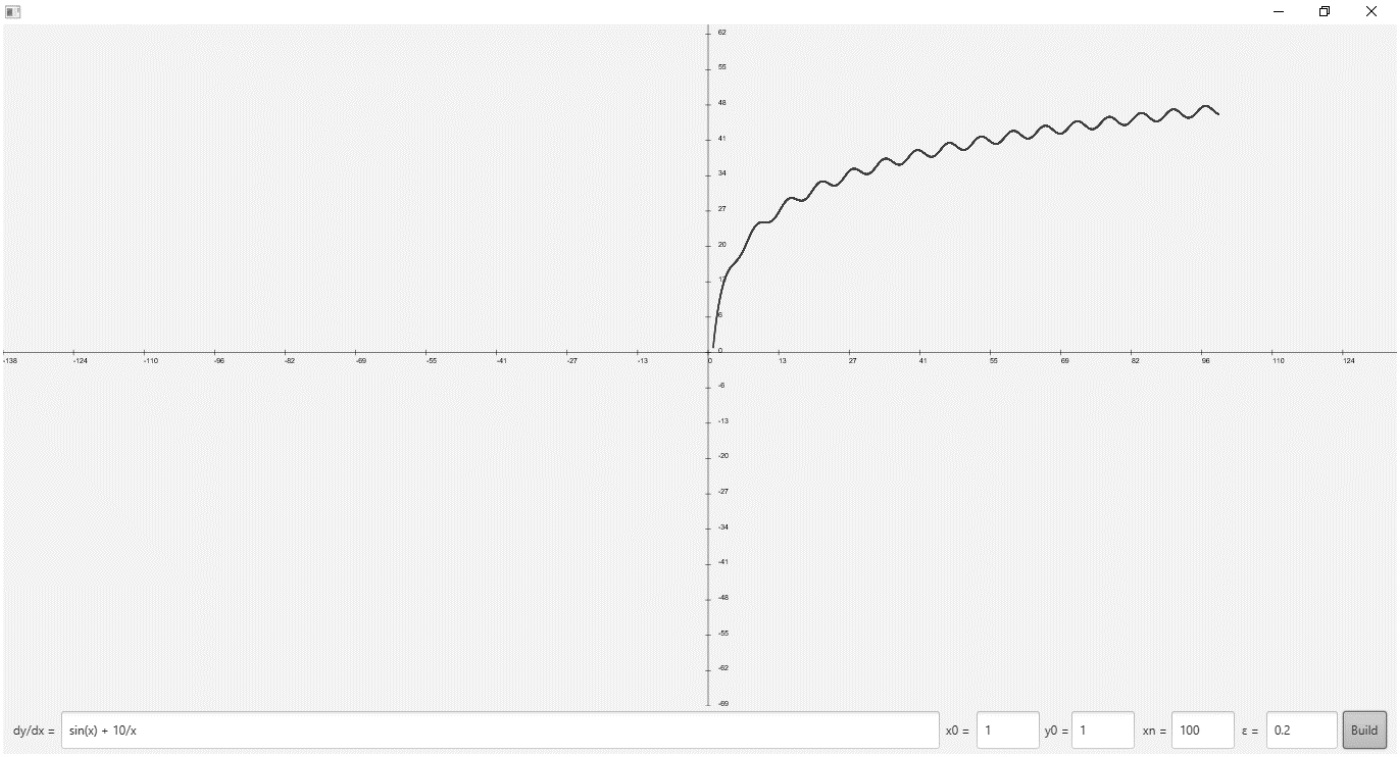
Код вспомогательного класса MethodRungeKutta не приведен

3. Примеры и результаты работы программы

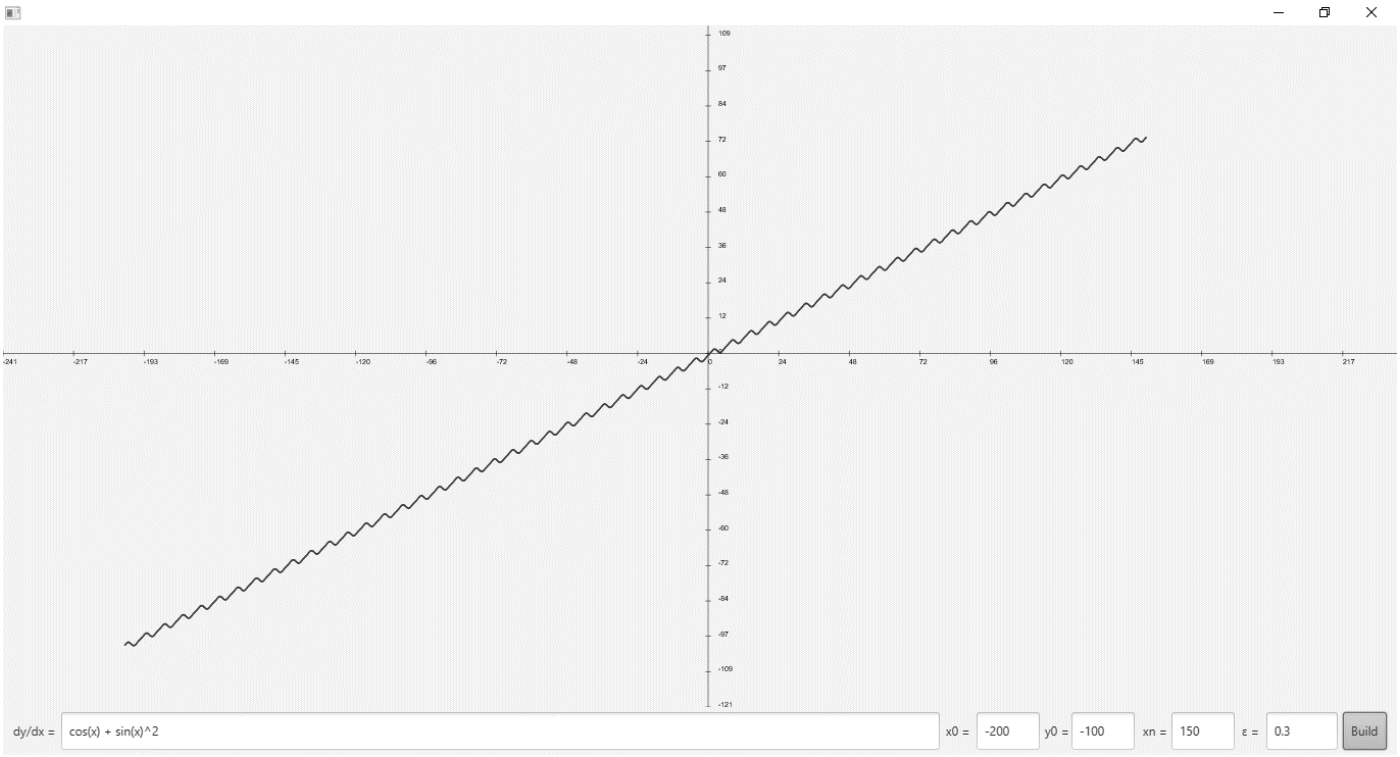
Пример 1.



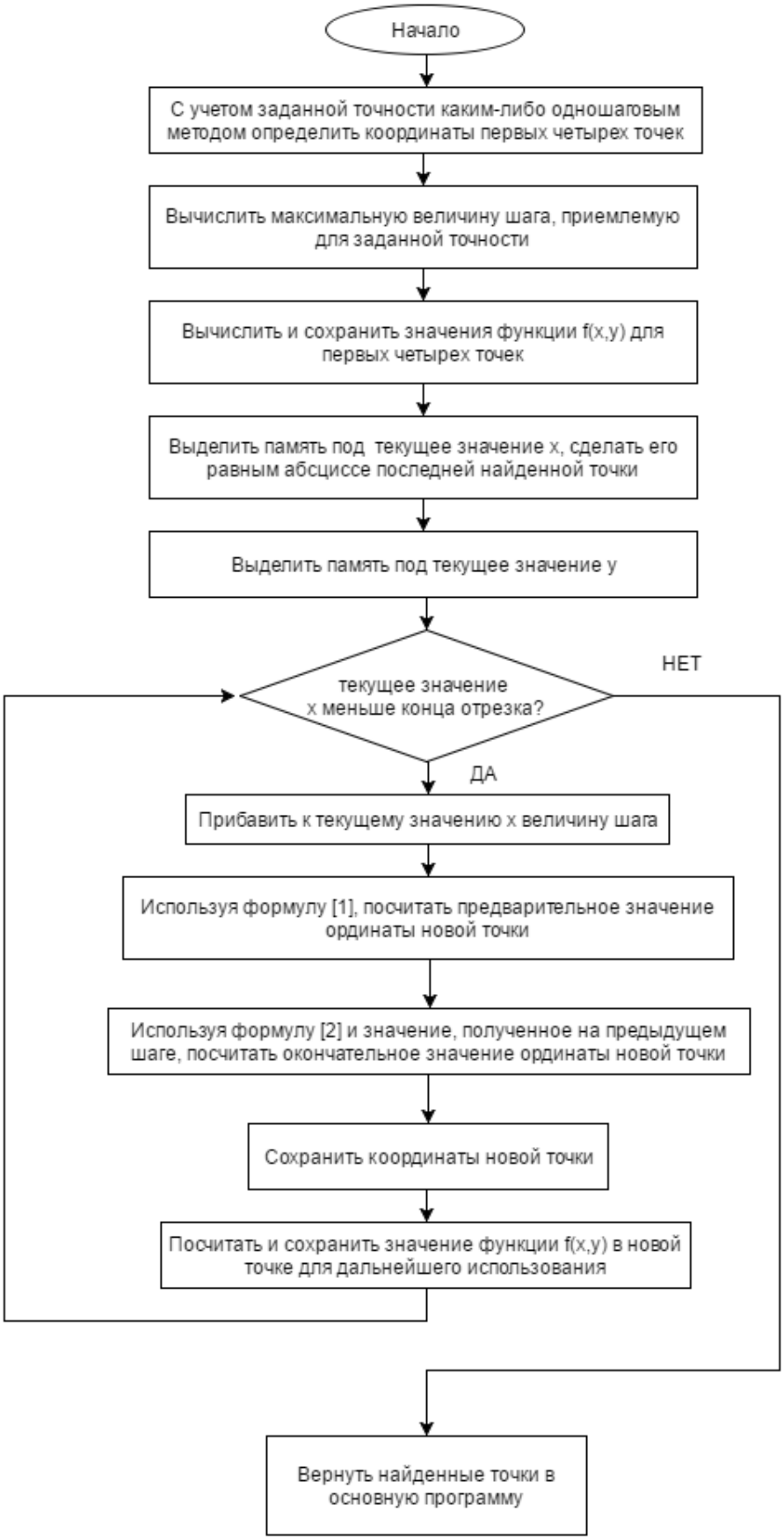
Пример 2.



Пример 3.



4. Блок – схема численного метода



5. Вывод

Таким образом, метод предиктора-корректора является более точным по сравнению с методом Адамса и модификациями метода Эйлера, и более экономичным по сравнению с методом Рунге-Кутта 4 порядка, поскольку требует вычисления всего 1 значения функции f(x,y) для каждой точки вместо четырех. В то же время метод предиктора-корректора требует наличия вспомогательных процедур для нахождения координат первых четырех точек, что несколько усложняет алгоритм. Рассмотренный метод также затрудняет изменение величины шага в процессе отыскания решения задачи Коши.