МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 6
Приближенное решение обыкновенных дифференциальных уравнений
Вариант №15

ОТЧЕТ

по лабораторной работе

по дисциплине

Вычислительная математика

РУКОВОДИТЕЛЬ:

________________ Суркова А.С.

СТУДЕНТ:

________________ Сапожников В.О.

19-ИВТ-3

Работа защищена «___» _____________

С оценкой _________________________

Нижний Новгород 2021

Содержание

1. Цель…………………………………………………………………..3
2. Постановка задачи……………………………………………………..4
3. Теоретические сведения…………………………………………………5
   3.1. Задача Коши………………………...……………………………...5
   3.2. Метод Эйлера и метод Эйлера с пересчетом……………………5
   3.3. Метод Рунге-Кутты………………………………………………..8
   3.4. Метод Адамса……………………………………………………10
4. Расчётные данные………………...………………………………….12
5. Листинг разработанной программы………………………………14
6. Результаты работы программы……………………………………24
7. Вывод………………………………………………………………….27

1. Цель

Закрепление знаний и умений по численному решению обыкновенных дифференциальных уравнений методом Эйлера, методом Эйлера с пересчетом, методом Рунге-Кутты и методом Адамса.

2. Постановка задачи

<u>*Задание 1.*</u>

Используя метод Эйлера и метод Эйлера с пересчетом, составить таблицу приближенных значений интеграла дифференциального уравнения $y' = f(x, y)$, удовлетворяющих начальным условиям $y(x_0) = y_0$ на отрезке $[a, b]$; шаг $h = 0.1$. Все вычисление вести с четырехзначными знаками. Проверить полученные значения, использую метод Рунге-Кутты 4 порядка.

$$15.\, y' = x + \sin\frac{y}{\pi} \qquad y_0(1.7) = 5.3, \;\; x \in [1.7; 2.7]$$

<u>*Задание 2.*</u>

Используя метод Адамса с третьими разностями составить таблицу приближенных значений интеграла дифференциального уравнения $y' = f(x, y)$, удовлетворяющих начальным условиям $y(x_0) = y_0$ на отрезке $[0, 1]$; шаг $h = 0.1$. Все вычисление вести с четырехзначными знаками. Начальный отрезок определить методом Рунге-Кутты. Проверить полученные значения, используя метод Эйлера с пересчетом.

$$15.\, y' = \cos(1.5x + y) + 1.5(x - y) \qquad y(0) = 0$$

3. Теоретические сведения

3.1. Задачи Коши

Требуется найти функцию $y = y(x)$, удовлетворяющую уравнению $y' = f(x, y)$ и принимающую при $x = x_0$ заданное значение y_0; $y(x_0) = y_0$. Считаем, что решение нужно найти для значения $x > x_0$. Производной функции $y = y(x)$ называется передел отношения приращения функции Δy к приращению аргумента Δx при стремлении последнего к нулю:

$$y' = f|x| = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

$\Delta x = h = 0.1$ в нашем случае.

3.2. Метод Эйлера и метод Эйлера с пересчетом

Рассмотрим уравнение $y' = f(x, y)$ в окрестности узлов $x = x_i (i = \overline{0..n})$ и заменим производную y' правой разностью $\frac{y_{i+1} - y_i}{h} = f(x, y)$. Полученная аппроксимация дифференциального уравнения $y' = f(x, y)$ имеет 1-ый порядок, поскольку при замене допускается погрешность $O(h)$. Выразим $y_{i+1} = y_i + hf(x_i, y_i)$. По этой формуле могут быть найдены значения сеточной функции в узлах. Построенный алгоритм называется методом Эйлера.

Погрешность.

$$y_i = y(x_i) - \delta_i$$

$$\delta_n = nO(h^2) = \frac{L}{h} O(h^2) = O(h)$$

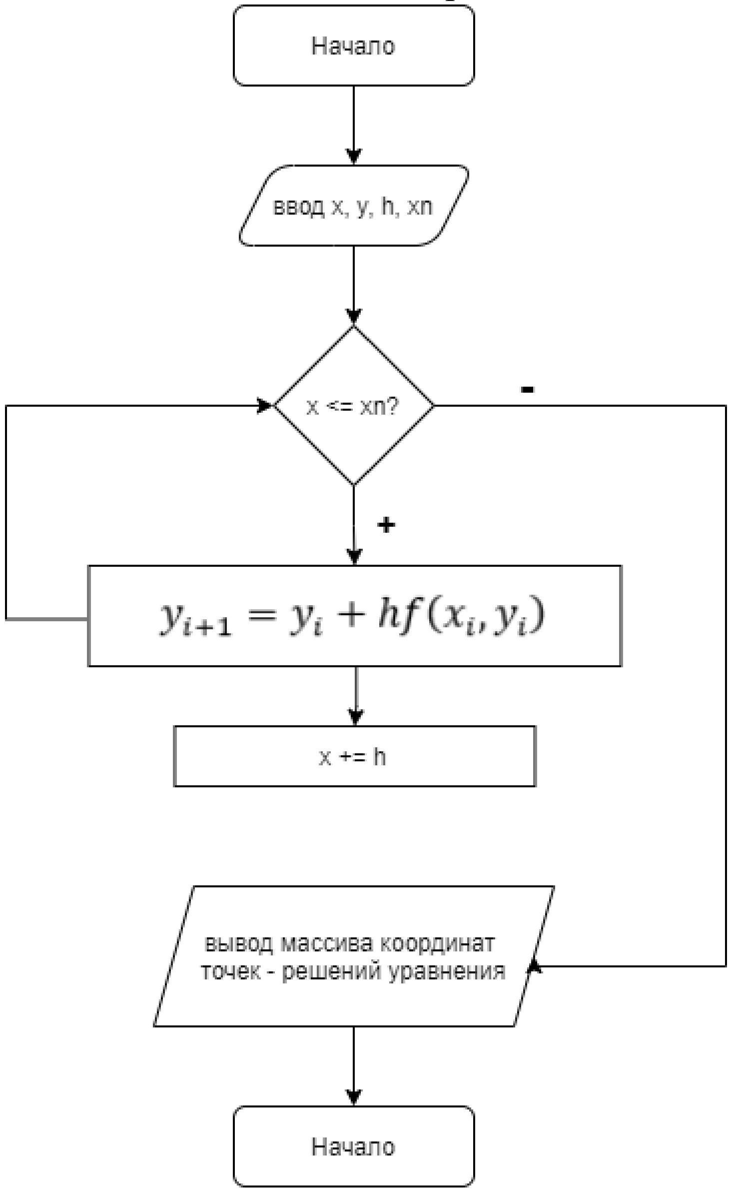
$$\delta_{i+1} = \delta_i + O(h^2)$$

Погрешность на каждом шаге увеличивается на $O(h^2)$.
Для повышения точности вычислений используют методом Эйлера с пересчетом. Однако для его реализации необходимо дважды вычислять правую часть функции. Заметим, что метод Эйлера с пересчетом представляет собой разновидность методом Рунге-Кутты.

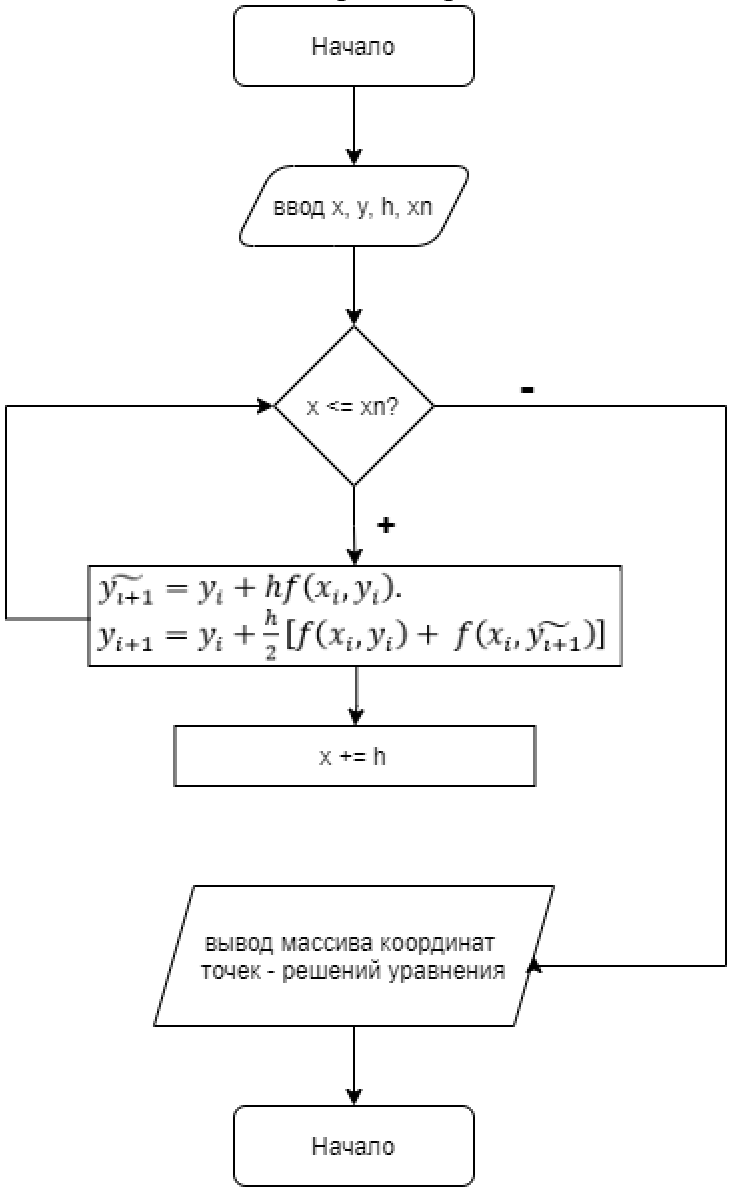
Первым действием найдем $\widetilde{y_{i+1}} = y_i + hf(x_i, y_i)$.

Вторым действием найдем $y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_i, \widetilde{y_{i+1}})]$

Метод Эйлера



Метод Эйлера с пересчетом



3.3. Метод Рунге-Кутты

Метод Рунге — Кутты четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты.

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка.

$y' = f(x, y), \qquad y(x_0) = y_0.$

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

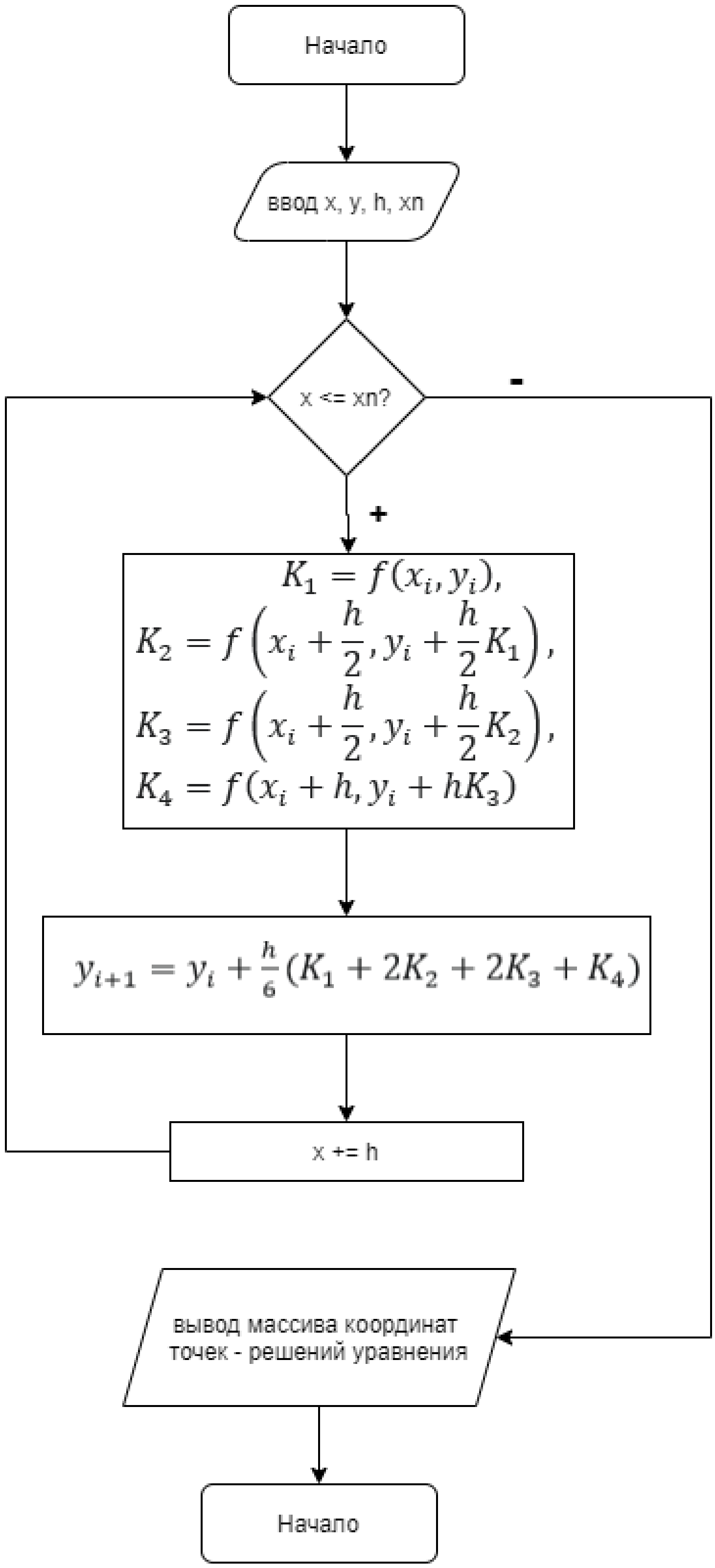
$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$

Вычисление нового значения проходит в четыре стадии:

$$K_1 = f(x_i, y_i),$$
$$K_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right),$$
$$K_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2\right),$$
$$K_4 = f(x_i + h, y_i + hK_3)$$

где h — величина шага сетки по x.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$.


Начало
ввод x, y, h, xn
x <= xn?
-
+
$K_1 = f(x_i, y_i),$
$K_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right),$
$K_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2\right),$
$K_4 = f(x_i + h, y_i + hK_3)$
$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$
x += h
вывод массива координат точек - решений уравнения
Начало

3.4. Метод Адамса

Запишем исходное уравнение $y' = f(x,y)$ в виде $dy = f(x,y)dx$.

Проинтегрируем обе части этого уравнения по x на отрезке $[x_i, x_{i+1}]$. Интеграл от левой части легко вычисляется: $\int_{x_i}^{x_{i+1}} dy(x) = y(x_{i+1}) - y(x_i) \approx y_{i+1} - y_i$

Для вычисления интеграла от правой части уравнения строится сначала интерполяционный многочлен P_{k-1} степени $k-1$ для аппроксимации функции $f(x,y)$ на отрезке $[x_i, x_{i+1}]$ по значениям f$(x_{i-k+1}, y_{i-k+1}), f(x_{i-k+2}, y_{i-k+2}), \dots, f(x_i, y_i)$. После этого можно написать:

$$\int_{x_i}^{x_{i+1}} f(x,y) \approx \int_{x_i}^{x_{i+1}} P_{k-1} dx$$

Приравнивая полученные выражение, можно получить формулу для определения неизвестного значения сеточной функции y_{i+1} в узле x_{i+1}:

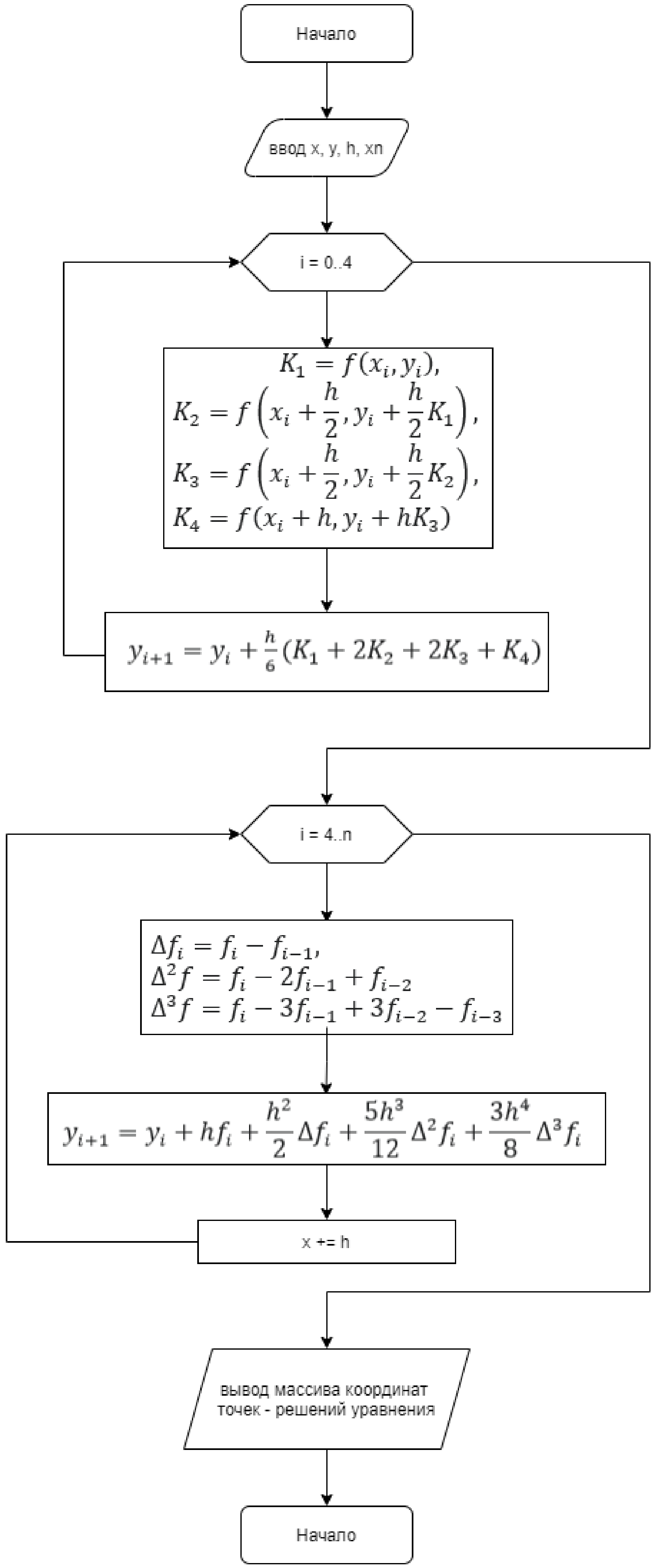
$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} P_{k-1} dx$$

На основе этой формулы можно строить различные многошаговые методы любого порядка точности, который зависит от степени интерполяционного многочлена $P_{k-1}(x)$. Методы Адамса широко распространены и при k=1 – получается метод Эйлера первого порядка. По условию нам нужен метод Адамса четвертого порядка, с третьими разностями. Для него необходимо 4 значения y и функции $f(x,y)$ предыдущих шагов. В качестве интерполяционного многочлена возьмем многочлен Ньютона. Конченые разности с учетом $h = const$:

$$\Delta f_i = f_i - f_{i-1},$$
$$\Delta^2 f = f_i - 2f_{i-1} + f_{i-2}$$
$$\Delta^3 f = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}$$

Тогда разностную схему четверного порядка метода Адамса можно записать после необходимых преобразований в следующем виде:

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i$$


Начало
ввод x, y, h, xn
i = 0..4
$K_1 = f(x_i, y_i),$
$K_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right),$
$K_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2\right),$
$K_4 = f(x_i + h, y_i + hK_3)$
$y_{i+1} = y_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$
i = 4..n
$\Delta f_i = f_i - f_{i-1},$
$\Delta^2 f = f_i - 2f_{i-1} + f_{i-2}$
$\Delta^3 f = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}$
$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i$
x += h
вывод массива координат точек - решений уравнения
Начало

4. Расчетные данные

Задание 1

Метод Эйлера

X	Y
1.7000	5.3000
1.8000	5.5693
1.9000	5.8473
2.0000	6.1331
2.1000	6.4259
2.2000	6.7249
2.3000	7.0291
2.4000	7.3377
2.5000	7.6498
2.6000	7.9647
2.7000	8.2817

Метод Эйлера с пересчетом

X	Y
1.7000	5.3000
1.8000	5.5686
1.9000	5.8455
2.0000	6.1299
2.1000	6.4288
2.2000	6.7174
2.3000	7.0190
2.4000	7.3246
2.5000	7.6334
2.6000	7.9448
2.7000	8.2580

Метод Рунге-Кутты

X	Y
1.7000	5.3000
1.8000	5.5730
1.9000	5.8542
2.0000	6.1429
2.1000	6.4382
2.2000	6.7392
2.3000	7.0450
2.4000	7.3548

2.5000	7.6678
2.6000	7.9831
2.7000	8.3003

Задание 2

Метод Адамса

X	Y
0.0000	0.0000
0.1000	0.0992
0.2000	0.1933
0.3000	0.2779
0.4000	0.3544
0.5000	0.4171
0.6000	0.4665
0.7000	0.5043
0.8000	0.5325
0.9000	0.5537
1.0000	0.5699

Метод Эйлера с пересчетом

X	Y
0.0000	0.0000
0.1000	0.0923
0.2000	0.1817
0.3000	0.2639
0.4000	0.3361
0.5000	0.3969
0.6000	0.4467
0.7000	0.4863
0.8000	0.5174
0.9000	0.5417
1.0000	0.5612

5. Листинг разработанной программы

Main.java

```
import function.Function;

import function.FirstFunction;
import function.SecondFunction;
import solution_strategy.*;

import java.util.Scanner;

/**
 * Класс, содержащий точку входа в программу - метод main.
 * Язык: java
 *
 * Реализация шестой лабораторной работы по дисциплине: Вычислительная математика
 * Вариант №15
 *
 * Текст задания:
 * Задание 1
 *  Используя метод Эйлера и метод Эйлера с пересчетом, составить
 *  таблицу приближенных значений интеграла дифференциального уравнения
 *  y´=f(x,y), удовлетворяющего начальным условиям y( на отрезке [a,b];
 *  шаг h=0.1. Все вычисления вести с четырехзначными знаками. Проверить
 *  полученные значения, используя метод Рунге-Кутты 4 порядка
 *
 *  Функция:
 *     y'=x+sin(y/pi)       X ∈ [1.7; 2.7]  y0(1.7) = 5.3
 *
 * @release: -     03.05.21
 * @last_update: - 03.05.21
 *
 * @author Vladislav Sapozhnikov 19-IVT-3
 */
public class Main
{
    //Константы для хранения последовательностей для
    //изменения цвета текста в консоли
    public static final String RESET = "\u001B[0m";
    public static final String PURPLE = "\u001B[35m";
    public static final String RED = "\u001B[31m";

    /**
     * Точка входа в программу
     * */
    public static void main(String[] args)
```

```
{
    System.out.println("\t\t\t\tЛабораторная работа №5 <<" + PURPLE + "Численное
       дифференцирование функций Ньютона и многочленом Лагранжа" + RESET + ">>");


    //Открытие потока ввода
    Scanner scanner = new Scanner(System.in);

    //Создание ссылки на объект, реализующий интерфейс
    //SolutionStrategy
    SolutionStrategy strategy = null;

    //Переменная для хранения результата ввода
    String ch = "";

    double x  = 0.0;
    double xn = 0.0;
    double y  = 0.0;

    Function function = null;

    //Выбор стратегии решения
    while (!ch.equals("q"))
    {
        System.out.println("Выберите способ:");
        System.out.println("\t1. Задание1. Метод Эйлера");
        System.out.println("\t2. Задание1. Метод Эйлера с пересчетом");
        System.out.println("\t3. Задание1. Метод Рунге-Кутта 4 порядка");
        System.out.println();
        System.out.println("\t4. Задание2. Метод Адамса");
        System.out.println("\t5. Задание2. Метод Эйлера с пересчетом");
        System.out.println();
        System.out.println("\tВведите q для выхода");
        System.out.print("Ввод: ");
        ch = scanner.nextLine();
        System.out.println();

        //Ввод с повторением
        switch (ch)
        {
            case ("1"):
            {
                strategy = new EulerSolution();
                function = new FirstFunction();
                x  = 1.7;
                xn = 2.7;
                y  = 5.3;
                break;
```

```
    }
    case ("2"):
    {
        strategy = new EulerRecalculationSolution();
        function = new FirstFunction();
        x  = 1.7;
        xn = 2.7;
        y  = 5.3;
        break;
    }
    case ("3"):
    {
        strategy = new RungeKuttSolution();
        function = new FirstFunction();
        x  = 1.7;
        xn = 2.7;
        y  = 5.3;
        break;
    }
    case ("4"):
    {
        strategy = new AdamsSolution();
        function = new SecondFunction();
        x  = 0.0;
        xn = 1.0;
        y  = 0;
        break;
    }
    case ("5"):
    {
        strategy = new EulerRecalculationSolution();
        function = new SecondFunction();
        x  = 0.0;
        xn = 1.0;
        y  = 0;
        break;
    }
    case ("q"):
            {
                System.out.println(RED + "Завершение работы..." + RESET);
                System.exit(0);
            }
    default:
    {
        System.out.println(RED + "Неверный ввод!" + RESET);
        break;
    }
}
```

```
            assert strategy != null;
            strategy.getSolution(function, x, xn, y, 0.1);
            System.out.println();
            System.out.println();
        }
    }
}
```

function/Function.java

```
package function;

/**
 * Интерфейс, содержащий основные методы функций.
 * */
public interface Function
{
    /**
     * Получение значения функции при заданных координатах.
     *
     * @param x - координата по x
     * @param y - координата по y
     * @return значение функции
     * */
    double getValue(double x, double y);
}
```

function/FirstFunction.java

```
package function;

/**
 * Класс - первая функция Варианта №15
 *
 * @see Function
 * */
public class FirstFunction implements Function
{
    /**
     * Получение значения функции при заданных координатах.
     *
     * @param x - координата по x
     * @param y - координата по y
     * @return значение функции
     * */
    @Override
    public double getValue(double x, double y)
    {
```

```
        return x + Math.sin(y / Math.PI);
    }
}
```

function/SecondFunction.java

```
package function;

/**
 * Класс - вторая функция Варианта №15
 *
 * @see Function
 * */
public class SecondFunction implements Function
{
    /**
     * Получение значения функции при заданных координатах.
     *
     * @param x - координата по x
     * @param y - координата по y
     * @return значение функции
     * */
    @Override
    public double getValue(double x, double y)
    {
        return Math.cos(1.5 * x + y) + 1.5 * (x - y);
    }
}
```

solution_strategy/SolutionStrategy.java

```
package solution_strategy;

import function.Function;

/**
 * Общий интерфейс всех стратегий решения.
 * */
public interface SolutionStrategy
{
    /**
     * Метод для получения массива точек являющихся решением
     * дифференциальных уравнений.
     * */
    void getSolution(Function function, double x, double xn, double y, double h);
}
```

solution strategy/EulerSolution.java

```
package solution_strategy;

import function.Function;

/**
 * Класс, реализующий решение простого дифференциального
 * уравнения методом Эйлера.
 *
 * @see SolutionStrategy
 * */
public class EulerSolution implements SolutionStrategy
{
    /**
     * Метод для получения массива точек, являющихся решением
     * дифференциальных уравнений при помощи метода Эйлера.
     *
     * @param function - уравнение значение которого необходимо
     *                   получить
     * @param x        - начальное значение x
     * @param xn       - конченое  значение x
     * @param y        - начальное значение y
     * @param h        - величина шага
     * */
    @Override
    public void getSolution(Function function, double x, double xn,
                                                 double y, double h)
    {
        //Добавляем в массив начальные условия
        System.out.printf("%.4f", x);
        System.out.print("  ");
        System.out.printf("%.4f", y);
        System.out.println();

        //по формуле высчитываем все остальные точки
        while (x <= xn)
        {
            y = y + h * function.getValue(x, y);
            x += h;

            System.out.printf("%.4f", x);
            System.out.print("  ");
            System.out.printf("%.4f", y);
            System.out.println();
        }
    }
}
```

solution_strategy/EulerRecalculationSolution.java

```
package solution_strategy;

import function.Function;

/**
 * Класс реализующий решение простого дифференциального
 * уравнения методом Эйлера с пересчетом.
 *
 * @see SolutionStrategy
 * */
public class EulerRecalculationSolution implements SolutionStrategy
{
    /**
     * Метод для получения массива точек, являющихся решением
     * дифференциальных уравнений при помощи метода Эйлера с пересчетом.
     *
     * @param function - уравнение значение которого необходимо получить
     * @param x        - начальное значение x
     * @param xn       - конченое  значение x
     * @param y        - начальное значение y
     * @param h        - величина шага
     * */
    @Override
    public void getSolution(Function function, double x, double xn, double y, double h)
    {
        double recalculationY;

        //Добавляем в массив начальные условия
        System.out.printf("%.4f", x);
        System.out.print("  ");
        System.out.printf("%.4f", y);
        System.out.println();

        //по формуле высчитываем все остальные точки
        int i = 1;
        while (x <= xn)
        {
            recalculationY = y + h * function.getValue(x, y);
            y = y + 0.5 * h * (function.getValue(x, y) + function.getValue(x,
recalculationY));
            x += h;

            System.out.printf("%.4f", x);
            System.out.print("  ");
            System.out.printf("%.4f", y);
            System.out.println();
```

```
            i++;
        }
    }
}
```

solution strategy/RungekuttSolution.java

```
package solution_strategy;

import function.Function;

/**
 * Класс реализующий решение простого дифференциального
 * уравнения методом Рунге-Кутта.
 *
 * @see SolutionStrategy
 * */
public class RungeKuttSolution implements SolutionStrategy
{
    /**
     * Метод для получения массива точек, являющихся решением
     * дифференциальных уравнений при помощи метода Рунге-Кутта.
     *
     * @param function - уравнение значение которого необходимо получить
     * @param x        - начальное значение x
     * @param xn       - конченое  значение x
     * @param y        - начальное значение y
     * @param h        - величина шага
     * */
    @Override
    public void getSolution(Function function, double x, double xn, double y, double h)
    {
        //Добавляем в массив начальные условия
        System.out.printf("%.4f", x);
        System.out.print("  ");
        System.out.printf("%.4f", y);
        System.out.println();

        double K1;
        double K2;
        double K3;
        double K4;

        //по формуле высчитываем все остальные точки
        while (x <= xn)
        {
            K1 = function.getValue(x , y);
```

```
            K2 = function.getValue(x + h / 4.0, y + (h / 4.0) * K1);
            K3 = function.getValue(x + h / 2.0, y + (h / 2.0) * K2);
            K4 = function.getValue(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3);

            y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0;
            x += h;

            System.out.printf("%.4f", x);
            System.out.print("  ");
            System.out.printf("%.4f", y);
            System.out.println();
        }
    }
}
```

solution_strategy/AdamsSolution.java

```
package solution_strategy;
import function.Function;
/**
 * Класс реализующий решение простого дифференциального
 * уравнения методом Адамса.
 *
 * @see SolutionStrategy
 * */
public class AdamsSolution implements SolutionStrategy
{
    /**
     * Метод для получения массива точек являющихся решением
     * дифференциальных уравнений при помощи метода Адамса.
     *
     * @param function - уравнение значение которго необходимо получить
     * @param x        - начальное значение x
     * @param xn       - конченое  значение x
     * @param y        - начальное значение y
     * @param h        - величина шага
     * */
    @Override
    public void getSolution(Function function, double x, double xn, double y, double h)
    {
        //Массив для хранения результатов
        double[][] res = new double[11][2];

        //Добавляем в массив начальные условия
        res[0][0] = x;
        res[0][1] = y;
        //Переменные для расчета методом Рунге-Кутта
        double K1;
```

```
        double K2;
        double K3;
        double K4;

        //Вычисляем начальный отрезок методом Рунге-Кутта
        for (int i = 1; i < 4; i++)
        {
            K1 = function.getValue(x , y);
            K2 = function.getValue(x + h / 4.0, y + (h / 4.0) * K1);
            K3 = function.getValue(x + h / 2.0, y + (h / 2.0) * K2);
            K4 = function.getValue(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3);

            y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0;
            x += h;

            res[i][0] = x;
            res[i][1] = y;

            System.out.printf("%.4f", res[i][0]);
            System.out.print("  ");
            System.out.printf("%.4f", res[i][1]);
            System.out.println();
        }

        //Вычисляем все остальные значения при помощи метода Адамса
        double[] df = new double[3];
        for (int i = 4; i < 11; i++)
        {
            df[0] = res[i][1] - res[i - 1][1];
            df[1] = res[i][1] - 2.0 * res[i - 1][1] + res[i - 2][1];
            df[1] = res[i][1] - 3.0 * res[i - 1][1] + 3.0 * res[i - 2][1] - res[i - 3][1];

            y = y + h * function.getValue(x, y) + (df[0] * h * h / 2.0) + 5.0 *
                        (df[1] * h * h * h / 12.0) + 3.0 * (df[2] * h * h * h * h / 8.0);

            x += h;

            res[i][0] = x;
            res[i][1] = y;

            System.out.printf("%.4f", res[i][0]);
            System.out.print("  ");
            System.out.printf("%.4f", res[i][1]);
            System.out.println();
        }
    }
}
```

6. Результаты работы программы

```
———— ———— ———— ————Лабораторная работа №5 <<Численное дифференцирование функций Ньютона и многочленом Лагранжа>>
Выберите способ:
————1. Задание1. Метод Эйлера
————2. Задание1. Метод Эйлера с пересчетом
————3. Задание1. Метод Рунге-Кутта 4 порядка

————4. Задание2. Метод Адамса
————5. Задание2. Метод Эйлера с пересчетом

————Введите q для выхода
Ввод: 1

1,7000  5,3000
1,8000  5,5693
1,9000  5,8473
2,0000  6,1331
2,1000  6,4259
2,2000  6,7249
2,3000  7,0291
2,4000  7,3377
2,5000  7,6498
2,6000  7,9647
2,7000  8,2817
Выберите способ:
————1. Задание1. Метод Эйлера
————2. Задание1. Метод Эйлера с пересчетом
————3. Задание1. Метод Рунге-Кутта 4 порядка

————4. Задание2. Метод Адамса
————5. Задание2. Метод Эйлера с пересчетом

————Введите q для выхода
Ввод: 2

1,7000  5,3000
1,8000  5,5686
1,9000  5,8455
2,0000  6,1299
2,1000  6,4208
2,2000  6,7174
2,3000  7,0190
2,4000  7,3246
2,5000  7,6334
2,6000  7,9448
2,7000  8,2580
```

```
Выберите способ:
—— 1. Задание1. Метод Эйлера
—— 2. Задание1. Метод Эйлера с пересчетом
—— 3. Задание1. Метод Рунге-Кутта 4 порядка

—— 4. Задание2. Метод Адамса
—— 5. Задание2. Метод Эйлера с пересчетом

—— Введите q для выхода
Ввод: 3

1,7000  5,3000
1,8000  5,5730
1,9000  5,8542
2,0000  6,1429
2,1000  6,4382
2,2000  6,7392
2,3000  7,0450
2,4000  7,3548
2,5000  7,6678
2,6000  7,9831
2,7000  8,3003
```

```
Выберите способ:
—— 1. Задание1. Метод Эйлера
—— 2. Задание1. Метод Эйлера с пересчетом
—— 3. Задание1. Метод Рунге-Кутта 4 порядка

—— 4. Задание2. Метод Адамса
—— 5. Задание2. Метод Эйлера с пересчетом

—— Введите q для выхода
Ввод: 4

0,1000  0,0992
0,2000  0,1933
0,3000  0,2779
0,4000  0,3544
0,5000  0,4171
0,6000  0,4665
0,7000  0,5043
0,8000  0,5325
0,9000  0,5537
1,0000  0,5699
```

```
—— Введите q для выхода
Ввод: 5

0,0000  0,0000
0,1000  0,0923
0,2000  0,1817
0,3000  0,2639
0,4000  0,3361
0,5000  0,3969
0,6000  0,4467
0,7000  0,4863
0,8000  0,5174
0,9000  0,5417
1,0000  0,5612
1,1000  0,5777


Выберите способ:
—— 1. Задание1. Метод Эйлера
—— 2. Задание1. Метод Эйлера с пересчетом
—— 3. Задание1. Метод Рунге-Кутта 4 порядка

—— 4. Задание2. Метод Адамса
—— 5. Задание2. Метод Эйлера с пересчетом

—— Введите q для выхода
Ввод: q

Завершение работы...
```

7. Вывод

В ходе данной работы были закреплены знания и умения по вычислению приближенных значений интеграла дифференциального уравнения при помощи методом Эйлера, Эйлера с пересчетом, Рунге-Кутты и Адамса.