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 № 5
Численное дифференцирование функций
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

Закрепление знаний и умений по численному дифференцированию функций с помощью интерполяционного многочлена Ньютона.

2. Постановка задачи

Вычислить первую и вторую производные функции в точках x, заданные таблицей.

15

x	y
3.50	33.1154
3.55	34.8133
3.60	36.5982
3.65	38.4747
3.70	40.4473
3.75	42.5211
3.80	44.7012
3.85	46.9931
3.90	49.4024
3.95	51.9354
4.00	54.5982
4.05	57.3975
4.10	60.3403
4.15	63.4340
4.20	66.6863

3. Теоретические сведения

Предположим, что функция $f(x)$, заданная в виде таблицы с постоянным шагом $h = x_i - x_{i-1}$ может быть аппроксимированная интерполяционным многочленом Ньютона:

$$y \approx N(x_0 + th) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0 \quad t = \frac{x - x_0}{h}$$

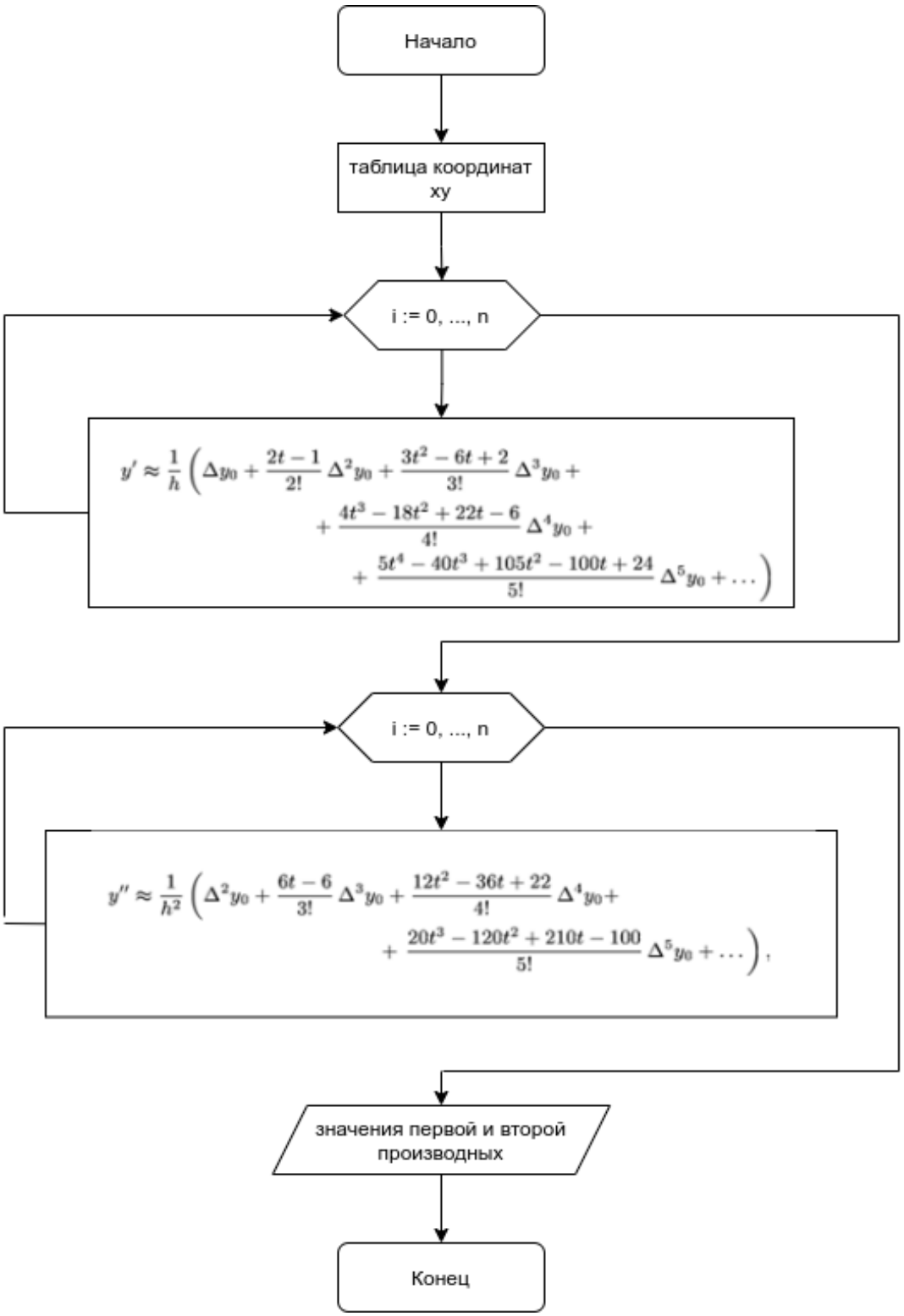
Дифференцируя этот многочлен по переменной x с учетом правила дифференцирования сложной функции:

$$\frac{dN}{dx} = \frac{dN}{dt}\frac{dt}{dx} = \frac{1}{h}\frac{dN}{dt},$$

можно получить формулы для вычисления производных любого порядка:

$$y' \approx \frac{1}{h}\left(\Delta y_0 + \frac{2t-1}{2!}\Delta^2 y_0 + \frac{3t^2-6t+2}{3!}\Delta^3 y_0 + \frac{4t^3-18t^2+22t-6}{4!}\Delta^4 y_0 + \frac{5t^4-40t^3+105t^2-100t+24}{5!}\Delta^5 y_0 + \ldots\right),$$

$$y'' \approx \frac{1}{h^2}\left(\Delta^2 y_0 + \frac{6t-6}{3!}\Delta^3 y_0 + \frac{12t^2-36t+22}{4!}\Delta^4 y_0 + \frac{20t^3-120t^2+210t-100}{5!}\Delta^5 y_0 + \ldots\right),$$


Начало
таблица координат xy
i := 0, ..., n
$y' \approx \frac{1}{h}\left(\Delta y_0 + \frac{2t-1}{2!}\Delta^2 y_0 + \frac{3t^2-6t+2}{3!}\Delta^3 y_0 + \frac{4t^3-18t^2+22t-6}{4!}\Delta^4 y_0 + \frac{5t^4-40t^3+105t^2-100t+24}{5!}\Delta^5 y_0 + \dots\right)$
i := 0, ..., n
$y'' \approx \frac{1}{h^2}\left(\Delta^2 y_0 + \frac{6t-6}{3!}\Delta^3 y_0 + \frac{12t^2-36t+22}{4!}\Delta^4 y_0 + \frac{20t^3-120t^2+210t-100}{5!}\Delta^5 y_0 + \dots\right),$
значения первой и второй производных
Конец

4. Расчетные данные

x	y'
3.50	33.1220
3.55	34.8118
3.60	36.5993
3.65	38.4755
3.70	40.4453
3.75	42.5728
3.80	44.7560
3.85	47.1680
3.90	49.8513
3.95	52.8545
4.00	56.2733
4.05	60.2168
4.10	64.7960
4.15	70.1438
4.20	76.4053

x	y''
3.50	32.6900
3.55	34.8267
3.60	36.6433
3.65	38.4200
3.70	40.4367
3.75	42.9733
3.80	46.3100
3.85	50.7267
3.90	56.5033
3.95	63.9200
4.00	73.2567
4.05	84.7933
4.10	98.8100
4.15	115.5867
4.20	135.4033

5. Листинг разработанной программы

Main.java

```
import solution_strategy.*;

import java.util.List;

/**
 * Класс, содержащий точку входа в программу - метод  main.
 * Язык: java
 *
 * Реализация пятой лабораторной работы по диспилине: Вычислительная математика
 * Вариант №15
 *
 * Текст задания:
 *  Найти первую и вторую производную функции в точках x, заданных
 *  таблицей, используя интерполяционные многочлены Ньютона. Сравнить со
 *  значениями производных, вычисленными по формулам, основанным на
 *  интерполировании многочленом Лагранжа (вычисление производных через
 *  значения функций).
 *
 *            x        y
 *          3.50    33.1154
 *          3.55    34.8133
 *          3.60    36.5982
 *          3.65    38.4747
 *          3.70    40.4473
 *          3.75    42.5211
 *          3.80    44.7012
 *          3.85    46.9931
 *          3.90    49.4024
 *          3.95    51.9354
 *          4.00    54.5982
 *          4.05    57.3975
 *          4.10    60.3403
 *          4.15    63.4340
 *          4.20    66.6863
 *
 *
 * @release: -     06.04.21
 * @last_update: - 06.04.21
 *
 * @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";

    /**
     * Точка входа в программу
     * */
    public static void main(String[] args)
    {
        System.out.println("\t\t\t\tЛабораторная работа №5 <<" + PURPLE + "«Численное
дифференцирование функций»" +
                RESET + ">>");

        //Таблица значений Вариант №15
        double[][] coordinates = {{3.50, 33.1154},
                                  {3.55, 34.8133},
                                  {3.60, 36.5982},
                                  {3.65, 38.4747},
                                  {3.70, 40.4473},
                                  {3.75, 42.5211},
                                  {3.80, 44.7012},
                                  {3.85, 46.9931},
                                  {3.90, 49.4024},
                                  {3.95, 51.9354},
                                  {4.00, 54.5982},
                                  {4.05, 57.3975},
                                  {4.10, 60.3403},
                                  {4.15, 63.4340},
                                  {4.20, 66.6863}};


        //Создание ссылки на объект, реализующий интерфейс
        //SolutionStrategy
        SolutionStrategy strategy = new NewtonDerivatives();

        //Получение 1ых производных
        List<Double> firstDerivatives  = strategy.getFirstDerivative(coordinates);

        //Получение 2ых производных
        List<Double> secondDerivatives = strategy.getSecondDerivative(coordinates);

        System.out.println();
        System.out.println("Заданная таблица координат: ");
        printCoordinates(coordinates);

        System.out.println("Значения первой производной: ");
        for (int i = 0; i < coordinates.length; i++)
        {
```

```
            System.out.printf("%.2f", coordinates[i][0]);
            System.out.print(" | ");
            System.out.printf("%.4f", firstDerivatives.get(i));
            System.out.println();
        }
        System.out.println();

        System.out.println("Значения второй производной: ");
        for (int i = 0; i < secondDerivatives.size(); i++)
        {
            System.out.printf("%.2f", coordinates[i][0]);
            System.out.print(" | ");
            System.out.printf("%.4f", secondDerivatives.get(i));
            System.out.println();
        }

    }

    /**
     * Метод для вывода таблицы заданных точек.
     *
     * @param coordinates   - двумерный массив значений [0][Xi] [1][Yi]
     * */
    public static void printCoordinates(double[][] coordinates)
    {
        System.out.println("     f(x)");
        System.out.println("  Xi |  =  Yi");
        for (double[] coordinate : coordinates)
        {
            System.out.printf("%.2f", coordinate[0]);
            System.out.print(" | ");
            System.out.printf("%.4f", coordinate[1]);
            System.out.println();
        }
        System.out.println();
    }
}
```

solution_strategy/SolutionStrategy.java

```
package solution_strategy;

import java.util.List;

/**
 * Общий интерфейс всех стратегий решения.
 *
 * @author Vladislav Sapozhnikov 19-IVT-3
```

```
 * @see NewtonDerivatives
 * */
public interface SolutionStrategy
{
    /**
     * Метод для получения первой производной различными способами.
     *
     * @param coordinates   - двумерный массив значений [0][Xi] [1][Yi]
     * @return список значений производной в заданных точках.
     * */
    List<Double> getFirstDerivative(double[][] coordinates);

    /**
     * Метод для получения второй производной различными способами.
     *
     * @param coordinates   - двумерный массив значений [0][Xi] [1][Yi]
     * @return список значений производной в заданных точках.
     * */
    List<Double> getSecondDerivative(double[][] coordinates);
}
```

solution_strategy/NewtonDerivates.java

```
package solution_strategy;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

/**
 * Класс, реализующий решение при помощи интерполяционной формулы Ньютона
 * для равноотстоящих узлов
 *
 * Реализует интерфейс SolutionStrategy
 *
 * @author Vladislav Sapozhnikov 19-IVT-3
 * @see SolutionStrategy
 * */
public class NewtonDerivatives implements SolutionStrategy
{
    /**
     * Вспомогательный метод для вывода конечных разностей i-ого порядка
     * в виде "лестничном виде"
     *
     * @param finiteDifferences - список списков конечных разностей
     * */
    private void printFiniteDifferences(List<List<Double>> finiteDifferences)
    {
```

```
    for (int i = 0; i < finiteDifferences.size(); i++)
    {
        //Если выводимое число будет меньше 10, то задаем дополнительный пробел
        //перед его выводом для равномерного отображения в консоли
        if (i < 9)
        {
            System.out.print("Конечные разности  " + (i+1) + " порядка: ");
        }
        else
        {
            System.out.print("Конечные разности " + (i+1) + " порядка: ");
        }

        for (int j = 0; j < finiteDifferences.get(i).size(); j++)
        {
            //Если число меньше 0 то выводим как оно есть,
            //иначе задаем дополнительный пробле для равномерного вывода в консоль
            if (finiteDifferences.get(i).get(j) < 0)
            {
                System.out.printf("%.5f", finiteDifferences.get(i).get(j));
            }
            else
            {
                System.out.printf(" %.5f", finiteDifferences.get(i).get(j));
            }
            System.out.print(" ");
        }
        System.out.println();
    }
}

/**
 * Вспомогательный метод для получения факториала
 *
 * @param n - число, от которого необходимо получить факториал
 *
 * @return - факториал переданного числа
 * */
private int getFact(int n)
{
    int res = 1;

    while (n > 1)
    {
        res *= n;
        n--;
    }
```

```
    return res;
}

/**
 * Вспомогательный метод для получения конечных разностей.
 *
 * @param coordinates - массив координат узлов интерполяции.
 * @return список списков конечных разностей
 * */
private List<List<Double>> getFiniteDifferences(double[][] coordinates)
{
    //Создание списка списков для хранения значений конченых разностей
    List<List<Double>> finiteDifferences = new LinkedList<>();

    //Ссылка на временный список для хранения промежуточных значений
    //конечных разностей
    List<Double> tempList;

    //В промежуточный список заносятся конечные разности 1-ого порядка
    tempList = new LinkedList<>();
    for (int i = 0; i < coordinates.length - 1; i++)
    {
        //Вычисление конечных разностей
        tempList.add(coordinates[i + 1][1] - coordinates[i][1]);
    }
    //промежуточный список ханосится в список списков кончеых разностей
    finiteDifferences.add(tempList);

    //На каждом i-ом шаге вычисляем значения конченых разностей нового порядка
    //и заносим в промежуточный список.
    //Полученный промежуточный список заносим в список списков промежуточных разностей
    for (int i = 0; i < coordinates.length-2; i++)
    {
        tempList = new LinkedList<>();    //инициализация промежуточного
                                          //списка пустым списком
        for (int j = 0; j < finiteDifferences.get(i).size() - 1; j++)
        {
            //Вычисление конечных разностей
            tempList.add(finiteDifferences.get(i).get(j + 1)
                                    - finiteDifferences.get(i).get(j));
        }
        finiteDifferences.add(tempList);
    }

    return finiteDifferences;
}

/**
```

```
 * Метод получения первой производной при помощи интерполяции многочленом
 * Ньютона.
 *
 * @param coordinates - массив координат узлов интерполяции.
 * @return список значений 1ой производной в заданных X-ах
 * */
@Override
public List<Double> getFirstDerivative(double[][] coordinates)
{
    //Получение конечных разностей и их вывод
    List<List<Double>> finiteDifferencesList = getFiniteDifferences(coordinates);
    printFiniteDifferences(finiteDifferencesList);

    List<Double> resList = new ArrayList<>();

    //Вычисление шага h
    double h = coordinates[1][0] - coordinates[0][0];

    double t;
    double res;

    for (double[] coordinate : coordinates)
    {
        //Вычисление параметра t, зависящего от h
        t = (coordinate[0] - coordinates[0][0]) / h;

        //к результату прибавляем Δy0
        res = finiteDifferencesList.get(0).get(0);

        //прибавляем к результату последующие слагаемые до Δ^5y0
        res += ((2.0 * t - 1.0) * finiteDifferencesList.get(1).get(0)
                / getFact(2));
        res += ((3.0 * t * t - 6.0 * t + 2.0) * finiteDifferencesList.get(2).get(0)
                / getFact(3));
        res += ((4.0 * t * t * t - 18.0 * t * t + 22.0 * t - 6.0) *
                  finiteDifferencesList.get(3).get(0) / getFact(4));

        res += ((5.0 * t * t * t * t - 40.0 * t * t * t + 105.0 * t * t - 100.0 * t +
                   24.0) * finiteDifferencesList.get(4).get(0) / getFact(5));

        //Делим полученный результат на h, т.к. узлы равноотстоящие
        res /= h;

        //Заносим результат в список ответов
        resList.add(res);
    }

    return resList;
```

```
    }

    /**
     * Метод получения второ1 производной при помощи интерполяции многочленом
     * Ньютона.
     *
     * @param coordinates - массив координат узлов интерполяции.
     * @return список значений 2ой производной в заданных X-ах
     * */
    @Override
    public List<Double> getSecondDerivative(double[][] coordinates)
    {
        //Получение конечных разностей и их вывод
        List<List<Double>> finiteDifferencesList = getFiniteDifferences(coordinates);

        List<Double> resList = new ArrayList<>();

        //Вычисление шага h
        double h = coordinates[1][0] - coordinates[0][0];
        double t;
        double res;

        for (double[] coordinate : coordinates)
        {
            //Вычисление параметра t, зависящего от h
            t = (coordinate[0] - coordinates[0][0]) / h;

            //к результату прибавляем Δ^2y0
            res = finiteDifferencesList.get(1).get(0);

            //прибавляем к результату последующие слагаемые до Δ^5y0
            res += ((6.0 * t - 6.0) * finiteDifferencesList.get(2).get(0))
                    / getFact(3);
            res += ((12.0 * t*t - 36.0 * t + 22.0) * finiteDifferencesList.get(3).get(0))
                    / getFact(4);
            res += ((20.0 *t*t*t - 120.0 * t*t + 210.0 * t -100.0)
 * finiteDifferencesList.get(4).get(0) / getFact(5));

            res /= (h * h);
            resList.add(res);
        }

        return resList;
    }

}
```

6. Результаты работы программы

```
Лабораторная работа №5 <<«Численное дифференцирование функций»>>
Конечные разности  1 порядка:  1,69790  1,78490  1,87650  1,97260  2,07380  2,18010  2,29190  2,40930  2,53300  2,66280  2,79930  2,94280  3,09370  3,25230
Конечные разности  2 порядка:  0,08700  0,09160  0,09610  0,10120  0,10630  0,11180  0,11740  0,12370  0,12980  0,13650  0,14350  0,15090  0,15860
Конечные разности  3 порядка:  0,00460  0,00450  0,00510  0,00510  0,00550  0,00560  0,00630  0,00610  0,00670  0,00700  0,00740  0,00770
Конечные разности  4 порядка: -0,00010  0,00060  0,00000  0,00040  0,00010  0,00070 -0,00020  0,00060  0,00030  0,00040  0,00030
Конечные разности  5 порядка:  0,00070 -0,00060  0,00040 -0,00030  0,00060 -0,00090  0,00080 -0,00030  0,00010 -0,00010
Конечные разности  6 порядка: -0,00130  0,00100 -0,00070  0,00090 -0,00150  0,00170 -0,00110  0,00040 -0,00020
Конечные разности  7 порядка:  0,00230 -0,00170  0,00160 -0,00240  0,00320 -0,00280  0,00150 -0,00060
Конечные разности  8 порядка: -0,00400  0,00330 -0,00400  0,00560 -0,00600  0,00430 -0,00210
Конечные разности  9 порядка:  0,00730 -0,00730  0,00960 -0,01160  0,01030 -0,00640
Конечные разности 10 порядка: -0,01460  0,01690 -0,02120  0,02190 -0,01670
Конечные разности 11 порядка:  0,03150 -0,03810  0,04310 -0,03860
Конечные разности 12 порядка: -0,06960  0,08120 -0,08170
Конечные разности 13 порядка:  0,15080 -0,16290
Конечные разности 14 порядка: -0,31370

Заданная таблица координат:
      f(x)
  Xi |  =  Yi
3,50 | 33,1154
3,55 | 34,8133
3,60 | 36,5982
3,65 | 38,4747
3,70 | 40,4473
3,75 | 42,5211
3,80 | 44,7012
3,85 | 46,9931
3,90 | 49,4024
3,95 | 51,9354
4,00 | 54,5982
4,05 | 57,3975
4,10 | 60,3403
4,15 | 63,4340
4,20 | 66,6863
Значения первой производной:
3,50 | 33,1220
3,55 | 34,8118
3,60 | 36,5993
3,65 | 38,4755
3,70 | 40,4453
3,75 | 42,5278
3,80 | 44,7560
3,85 | 47,1768
3,90 | 49,8513
3,95 | 52,8545
4,00 | 56,2753
4,05 | 60,2168
4,10 | 64,7960
4,15 | 70,1438
4,20 | 76,4053

Значения второй производной:
3,50 | 32,6900
3,55 | 34,8267
3,60 | 36,6433
3,65 | 38,4200
3,70 | 40,4367
3,75 | 42,9733
3,80 | 46,3100
3,85 | 50,7267
3,90 | 56,5033
3,95 | 63,9200
4,00 | 73,2567
4,05 | 84,7933
4,10 | 98,8100
4,15 | 115,5867
4,20 | 135,4033
```

7. Вывод

В ходе данной работы были закреплены знания и умения по вычислению производных первого и второго порядка при помощи интерполяционного многочлена Ньютона.