Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A picture containing logo

Description automatically generated

Лабораторная работа №5 по дисциплине  
«Вычислительная математика»

" Интерполяция функции"

Выполнил: Дау Конг Туан Ань

Группа: P32151

Преподаватель: Машина Е.А

г. Санкт-Петербург

2023

Вычислительная реализация задачи:

1. Выбрать из табл.

1 заданную по варианту таблицу (таблица 1.1 – таблица 1.5);

2. Построить таблицу конечных разностей для заданной таблицы. Таблицу отразить в отчете;

3. Вычислить значения функции для аргумента (см. табл.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться;

4. Вычислить значения функции для аргумента (см. табл. 1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться;

5. Подробные вычисления привести в отчете.

Table

Description automatically generated

2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | 3.7587 | 0.4274 | 0.3083 | -0.6171 | 1.0778 | -1.7774 | 2.9759 |
|  | 4.1861 | 0.7357 | -0.3088 | 0.4607 | -0.6996 | 1.1983 |  |
|  | 4.9218 | 0.4269 | 0.1519 | -0.2389 | 0.4987 |  |  |
|  | 5.3487 | 0.5788 | -0.087 | 0.2598 |  |  |  |
|  | 5.9275 | 0.4918 | 0.1728 |  |  |  |  |
|  | 6.4193 | 0.6646 |  |  |  |  |  |
|  | 7.0839 |  |  |  |  |  |  |

3.

Graphical user interface, text, application, Word

Description automatically generated

*4.*

*Text

Description automatically generated*

Text

Description automatically generated

Table

Description automatically generated

**Gauss:**

public Double predictValueAt(Double x) {  
 int iter = 0;  
 List<Point> pointList = this.inputSet.getPointList();  
 for(int i = 1 ;i < pointList.size(); ++i) {  
 if(Math.*abs*(x - pointList.get(i).getX()) <  
 Math.*abs*(x - pointList.get(iter).getX()))  
 iter = i;  
 }  
  
 Double h = Math.*abs*((pointList.get(1).getX() - pointList.get(0).getX()));  
 if((x - pointList.get(iter).getX()) > 0)  
 return firstCase(iter, (x - pointList.get(iter).getX())/h);  
 return secondCase(iter, (x - pointList.get(iter).getX()) / h);  
  
}  
  
public Double firstCase(Integer iter, Double t) {  
 double res = finiteDiffTable[iter][0] +  
 finiteDiffTable[iter][1]  
 \* t;  
 for(int i = 2 ;i < this.inputSet.getSize(); i+= 1) {  
 if(i %2 == 0) --iter;  
 if(iter < 0) return res;  
 Double s = 0.0;  
 int temp = 1;  
 for(int j = -i/2 ;j < i/2; ++j) {  
 s \*= (t + j);  
 s /= temp++;  
 }  
 s\*=finiteDiffTable[iter][i];  
 res += s;  
 }  
  
 return res;  
}  
  
public Double secondCase(Integer iter, Double t) {  
 if(iter == 0) return finiteDiffTable[iter][0];  
 double res = finiteDiffTable[iter][0] +  
 finiteDiffTable[--iter][1]  
 \* t;  
 for(int i = 2 ;i < this.inputSet.getSize(); i+= 1) {  
 if(i % 2 == 1) --iter;  
 if(iter < 0) return res;  
 Double s = 0.0;  
 int temp = 1;  
 for(int j = -i/2 ;j < i/2; ++j) {  
 s \*= (t + j);  
 s /= temp++;  
 }  
 s\*=finiteDiffTable[iter][i];  
 res += s;  
 }  
  
 return res;

**Lagrange:**

List<Point> pointList = this.inputSet.getPointList();  
for(int i = 0 ;i < pointList.size(); ++i) {  
 if(pointList.get(i).getX() == x) {  
 return pointList.get(i).getY();  
 }  
}  
Double mulAllX = 1.0;  
Double res = 0.0;  
for(int i = 0 ;i < pointList.size();++i) {  
 mulAllX \*= (x - pointList.get(i).getX());  
}  
for(int i =0 ; i < pointList.size(); ++i) {  
 Double temp = mulAllX / (x - pointList.get(i).getX());  
 for(int j = 0; j < i ; ++j) {  
 temp /= (pointList.get(i).getX() - pointList.get(j).getX());  
 }  
  
 for(int j = i + 1; j < pointList.size() ; ++j) {  
 temp /= (pointList.get(i).getX() - pointList.get(j).getX());  
 }  
  
 temp \*= pointList.get(i).getY();  
 res += temp;  
}  
  
return res;

C:\Users\Never\.jdks\openjdk-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1\lib\idea\_rt.jar=56857:C:\Program Files\JetBrains\IntelliJ IDEA 2023.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Never\IdeaProjects\Lab5\_CM\out\production\Lab5\_CM;C:\Users\Never\.m2\repository\org\jfree\jfreechart\1.5.4\jfreechart-1.5.4.jar Main

Choose way to import input set:

1 : By console

2 : By file

3 : By function

2

Choose algorithm:

1 : Lagrange

2 : Gauss

2

Finite different table:

3.7587 0.4273 0.3084 -0.6173 1.0781 -1.7778 2.9762

4.1861 0.7357 -0.3089 0.4608 -0.6997 1.1984 0.0

4.9218 0.4268 0.1519 -0.2389 0.4987 0.0 0.0

5.3487 0.5788 -0.087 0.2598 0.0 0.0 0.0

5.9275 0.4917 0.1728 0.0 0.0 0.0 0.0

6.4193 0.6646 0.0 0.0 0.0 0.0 0.0

7.0839 0.0 0.0 0.0 0.0 0.0 0.0

Type point:

2.112

Result: 3.8612

Chart, line chart

Description automatically generated

Вывод: полином Лагранжа дает большую ошибку, чем полином Ньютона или Гаусса, но, в свою очередь, полином Лагранжа дает более общую формулу. В общем, все три полинома дают большую сложность