**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»**

**Факультет программной инженерии и компьютерной техники**

**Вычислительная математика**

**Лабораторная работа №5**

**Вариант 8**

Группа: P3266

Выполнили:

Тажибаева Е.В Коляда А.А

Проверила:

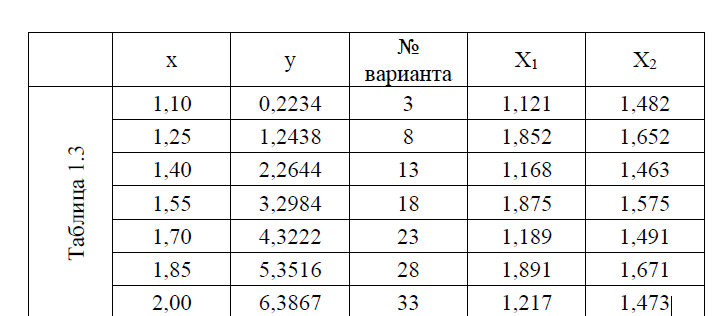
Машина Е. А.

Г. Санкт-Петербург

**Цель лабораторной работы**

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

**Вычислительная часть**



Метод Ньютона

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **xi** | **yi** | **Δyi** | **Δ^2yi** | **Δ^3yi** | **Δ^4yi** | **Δ^5yi** | **Δ^6yi** |
| **x0** | 1,1 | 0,2234 | 1,0204 | 0,0002 | 0,0132 | -0,0368 | 0,0762 | -0,1313 |
| **x1** | 1,25 | 1,2438 | 1,0206 | 0,0134 | -0,0236 | 0,0394 | -0,0551 |  |
| **x2** | 1,4 | 2,2644 | 1,034 | -0,0102 | 0,0158 | -0,0157 |  |  |
| **x3** | 1,55 | 3,2984 | 1,0238 | 0,0056 | 0,0001 |  |  |  |
| **x4** | 1,7 | 4,3222 | 1,0294 | 0,0057 |  |  |  |  |
| **x5** | 1,85 | 5,3516 | 1,0351 |  |  |  |  |  |
| **x6** | 2 | 6,3867 |  |  |  |  |  |  |

Метод Гаусса x>a

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **xi** | **yi** | **Δyi** | **Δ^2yi** | **Δ^3yi** | **Δ^4yi** | **Δ^5yi** | **Δ^6yi** |
| **x-3** | 1,1 | 0,2234 | 1,0204 | 0,0002 | 0,0132 | -0,0368 | 0,0762 | -0,1313 |
| **x-2** | 1,25 | 1,2438 | 1,0206 | 0,0134 | -0,0236 | 0,0394 | -0,0551 |  |
| **x-1** | 1,4 | 2,2644 | 1,034 | -0,0102 | 0,0158 | -0,0157 |  |  |
| **x0** | 1,55 | 3,2984 | 1,0238 | 0,0056 | 0,0001 |  |  |  |
| **x1** | 1,7 | 4,3222 | 1,0294 | 0,0057 |  |  |  |  |
| **x2** | 1,85 | 5,3516 | 1,0351 |  |  |  |  |  |
| **x3** | 2 | 6,3867 |  |  |  |  |  |  |

**Листинг программы**

Многочлен Лагранжа

public class L5Method1 implements IL5Method, IFunction {  
 @Override  
 public String toString() {  
 return "Многочлен Лагранжа";  
 }  
 @Override  
 public double method(double[] xs, double[] ys, double x) {  
 setXY(xs,ys);  
 return functionOf(x);  
 }  
  
 private double[] xs;  
 private double[] ys;  
  
 @Override  
 public void setXY(double[] xs, double[] ys) {  
 this.xs = xs;  
 this.ys = ys;  
 }  
  
 @Override  
 public double functionOf(double x) {  
 double result = 0;  
 for(int i = 0; i < xs.length; i++){  
 double mlt = 1;  
 for(int j = 0; j < ys.length; j++){  
 if(j==i) continue;  
 mlt \*= (x - xs[j])/(xs[i] - xs[j]);  
 }  
 result += mlt\*ys[i];  
 }  
 return result;  
 }  
}

Многочлен Ньютона с разделенными разностями

public class L5Method2 implements IL5Method, IFunction {  
 @Override  
 public double method(double[] xs, double[] ys, double x) {  
 setXY(xs,ys);  
 return functionOf(x);  
 }  
  
 private double[] xs;  
 private double[] ys;  
 @Override  
 public void setXY(double[] xs, double[] ys) {  
 this.xs = xs;  
 this.ys = ys;  
 }  
  
 */\*\*  
 \* Расчет разделенных разностей  
 \* @param iStart начальный элемент  
 \* @param iLast конечный элемент  
 \* @return конечная разность  
 \*/* private double dividedDiff(int iStart, int iLast) {  
 if((iLast - iStart) == 1){  
 return (ys[iLast] - ys[iStart])/(xs[iLast] - xs[iStart]);  
 }  
 else {  
 return (dividedDiff(iStart + 1, iLast)  
 - dividedDiff(iStart,iLast-1)) / (xs[iLast] - xs[iStart]);  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Многочлен Ньютона с разделенными разностями";  
 }  
  
 @Override  
 public double functionOf(double x) {  
 double result = ys[0];  
 for (int k = 1; k < xs.length; k++) {  
 double divdiff = dividedDiff(0,k);  
 double mlt = 1;  
 for(int j = 0; j < k; j++){  
 mlt \*= (x - xs[j]);  
 }  
 result += divdiff\*mlt;  
 }  
 return result;  
 }  
}

Многочлен Ньютона для равноотстоящих углов

public class L5Method3 implements IL5Method, IFunction {  
  
 private double [] xs;  
 private double [] ys;  
 */\*\*  
 \* Шаг  
 \*/* private double h;  
  
 */\*\*  
 \* Строит таблицу конечных разностей размером n на n+ (x + n колонок для delta)  
 \* @return  
 \*/* public double[][] getFinalDiffTable(){  
 double[][] diffTable = new double[xs.length][xs.length+1];  
 for(int i = 0; i < xs.length; i++){  
 diffTable[i][0] = xs[i];  
 for(int j = 0; j < xs.length-i; j++){  
 diffTable[i][j+1] = finalDiff(i,j);  
 }  
 }  
 return diffTable;  
 }  
  
 */\*\*  
 \* Метод вычисляет конучную разность к-го порядка для i-го элемента  
 \* @param i номер узла интерполяции  
 \* @param k порядок конченой разности  
 \* @return величина конечной разности  
 \*/* private double finalDiff(int i,int k) {  
 if(k == 0){  
 return ys[i];  
 } else {  
 return finalDiff(i+1, k-1) - finalDiff( i, k-1);  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "Многочлен Ньютона с конечными разностями";  
 }

Многочлен Гаусса

public class L5Method4 implements IL5Method, IFunction {  
 protected double xs[];  
 protected double ys[];  
 */\*\*  
 \* Шаг  
 \*/* private double h;  
 */\*\*  
 \* Величина t - (x - a)/h  
 \*/* protected double t;  
 */\*\*  
 \* Параметр n  
 \*/* protected int n;  
 */\*\*  
 \* Индекс в массиве выбранной центральной точки  
 \*/* private int xi0;  
  
 @Override  
 public String toString() {  
 return "Многочлен Гаусса";  
 }  
  
 */\*\*  
 \*  
 \* @param x функция от  
 \* @return  
 \*/* @Override  
 public double functionOf(double x) {  
 return switch(calcMiddle(x)){  
 case 1 -> function1(x);  
 default -> function2(x);  
 };  
 }  
  
 */\*\*  
 \* Расчет по первой формуле Гаусса  
 \* @param x  
 \* @return  
 \*/* private double function1(double x){  
 double res = 0;  
 int k = 0; // Порядок конечной разницы и онже номер элемента многочлена  
 double tVal = 1;  
 for(int i = 0; i >= -n; i--){  
 tVal = calcT1(tVal,k);  
 double dif = finalDiff(i,k++);  
 res += tVal \* dif;  
  
 if(k > 2\*n) break;  
 tVal = calcT1(tVal,k);  
 dif = finalDiff(i,k++);  
 res += tVal \* dif;  
 }  
 return res;  
 }  
 private double function2(double x){  
 double res = ys[iToIndex(0)];  
 int k = 1; // Порядок конечной разницы и онже номер элемента многочлена  
 double tVal = 1;  
 for(int i = -1; i >= -n; i--){  
 tVal = calcT2(tVal,k);  
 double dif = finalDiff(i,k++);  
 res += tVal \* dif;  
  
 if(k > 2\*n) break;  
 tVal = calcT2(tVal,k);  
 dif = finalDiff(i,k++);  
 res += tVal \* dif;  
 }  
 return res;  
 }  
  
  
 */\*\*  
 \* Вычисление множителя t по первой формуле Гаусса для k-го элемента  
 \* @param k номер элемента многочлена  
 \* @return  
 \*/* private double calcT1(double lastT,int k){  
 /\* toMinus  
 при k = 1 получим 0  
 при k = 2 получим -1  
 при k = 3 получим 1  
 при k = 4 получим -2  
 при k = 5 получим 2  
 \*/  
 if(k == 0) return 1;  
 int toMinus = k/2 \* pow1(k-1);  
 double tres = lastT \* (t + toMinus)/k;  
 return tres;  
 }  
 */\*\*  
 \* Вычисление множителя t по второй формуле Гаусса для k-го элемента  
 \* @param k номер элемента многочлена  
 \* @return  
 \*/* private double calcT2(double lastT,int k){  
 /\* toMinus -  
 при k = 1 получим 0  
 при k = 2 получим 1  
 при k = 3 получим -1  
 при k = 4 получим 2  
 при k = 5 получим -2  
 \*/  
 if(k == 0) return 1;  
 int toMinus = k/2 \* pow1(k);  
 double tres = lastT \* (t + toMinus)/k;  
 return tres;  
 }  
  
 */\*\*  
 \* Рассчитывает -1 в степени k  
 \* @param k - степень  
 \* @return -1 в степени k  
 \*/* private int pow1(int k){  
 if(k == 0) return 1;  
 return -1 \* pow1(k-1);  
 }  
  
 @Override  
 public double method(double[] xs, double[] ys, double x) {  
 setXY(xs,ys);  
 return functionOf(x);  
 }  
  
 @Override  
 public void setXY(double[] xs, double[] ys) {  
 this.xs = xs;  
 this.ys = ys;  
 if(xs.length != ys.length || xs.length < 2)  
 throw new CalcErrorException("Недостаточное количество точек.");  
 h = Math.*abs*(xs[1] - xs[0]);  
 }  
  
 */\*\*  
 \* Метод определяет нулевую точку в массиве  
 \* @param x  
 \*/* protected int calcMiddle(double x) {  
  
 xi0 = xs.length%2 == 0? (xs.length)/2-1: xs.length/2;  
 n = xi0;  
 t =(x - xs[xi0])/h;  
  
 return x > xs[xi0] ? 1 : 2;  
  
 }  
  
 */\*\*  
 \* Метод вычисляет конучную разность к-го порядка для i-го элемента  
 \* @param i номер узла интерполяции  
 \* @param k порядок конченой разности  
 \* @return величина конечной разности  
 \*/* protected double finalDiff(int i,int k) {  
 if(k == 0){  
 return ys[iToIndex(i)];  
 } else {  
 return finalDiff(i+1, k-1) - finalDiff(i, k-1);  
 }  
 }  
  
 protected int iToIndex(int i){  
 int index = xi0+i;  
 if(index < 0 || index >= xs.length) throw new CalcErrorException("Индекс при расчете вышел за пределы массива узлов интерполяции");  
 return xi0 + i;  
 }  
}

Многочлен Стирлинга

public class L5Method5 extends L5Method4{  
  
 @Override  
 public double functionOf(double x) {  
 calcMiddle(x);  
  
 double res = ys[iToIndex(0)];  
  
 int k = 1; // Номер члена в многочлене  
 double t1 = 1; // Текущее значение коэффициента t\*(t^2-1^2)....  
 double tf = 1; // 1/f  
 t1 =1;  
 k = 1;  
 tf = 1;  
 double tt = t; // используется при i =1  
 double tt2 = t\*t; // В дальнейшем используем t в квадрате  
 for(int i = 1; i <= n; i++){  
 if(i> 1) tt = tt2;  
  
 tf = tf/k; // в начале 1,для к = 3 будет 1/6  
 t1 \*= (tt - (i-1)\*(i-1)); // в начале t, для k = 3 будет е\*(  
 res += tf \* t1 \* (finalDiff(-i,k) + finalDiff(-i+1,k))/2;  
 k++;  
 tf = tf/k; //В начале 1/2 на второй итерации 1/fac(4)  
 res += t\*tf \*t1 \* finalDiff(-i,k);  
 k++;  
 }  
 return res;  
 }

Многочлен Бесселя

public class L5Method6 extends L5Method4{  
 @Override  
 public double functionOf(double x) {  
 calcMiddle(x);  
  
 double tMinusHalf = t -1/2.;  
 double res = (ys[iToIndex(0)] + ys[iToIndex(1)])/2 + tMinusHalf\*finalDiff(0,1);  
  
 int k = 1; // Номер члена в многочлене  
 double t1 = t; // Текущее значение коэффициента t\*(t^2-1^2)....  
 double tf = 1; // 1/f  
 for(int i = 1; i <= n; i++){  
 k++;  
 if(i == 1)  
 t1 = t1 \*(t - 1);  
 else {  
 t1 = t1 \*(t - i)\*(t+i-1);  
 }  
 tf = tf/k;  
 res += t1 \* tf \* (finalDiff(-i,k) + finalDiff(-i+1,k))/2;  
 k++;  
  
 tf=tf/k;  
 if(tMinusHalf != 0) res += tMinusHalf \* t1 \* tf \* finalDiff(-i,k);  
  
 }  
 return res;  
 }

**Результаты работы программы**

**Изображение выглядит как текст, снимок экрана, число, диаграмма

Автоматически созданное описание**

**Вывод**

В ходе лабораторной работы была решена задача интерполяции для табличной функции с использованием методов Ньютона и Гаусса.

Метод Ньютона:

• Для вычисления значения функции при использовалась первая интерполяционная формула Ньютона, так как находится ближе к началу интерполяционного узла.

• В результате интерполяции получено значение

Метод Гаусса:

• Для вычисления значения функции при использовалась вторая интерполяционная формула Гаусса, так как находится ближе к концу интерполяционного узла.

• В результате интерполяции получено значение

В ходе лабораторной работы были успешно применены методы Ньютона и Гаусса для интерполяции табличной функции. Вычисленные значения функции для заданных аргументов x\_1 и x\_2 показывают, что методы интерполяции позволяют получить точные результаты даже при отсутствии значений функции в заданных точках.