**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кафедра вычислительной математики**

Лабораторная работа № 1

«Приближении функции с помощью

интерполяционного полинома Лагранжа»

Выполнила: Шелег Владислава Михайловна

2 курс, 9 группа

Преподаватель: Никифоров И. В

МИНСК

2017

**Постановка задачи**

Пусть дана функция в данном варианте . Необходимо построить интерполяционный полином Лагранжа -ой степени по узлам равномерной сетки узлов.

**Алгоритм решения**

Лагранж предложил способ вычисления многочленов вида:

где базисные полиномы определяются по формуле:

где есть -ое значение функции в точке.

обладают следующими свойствами:

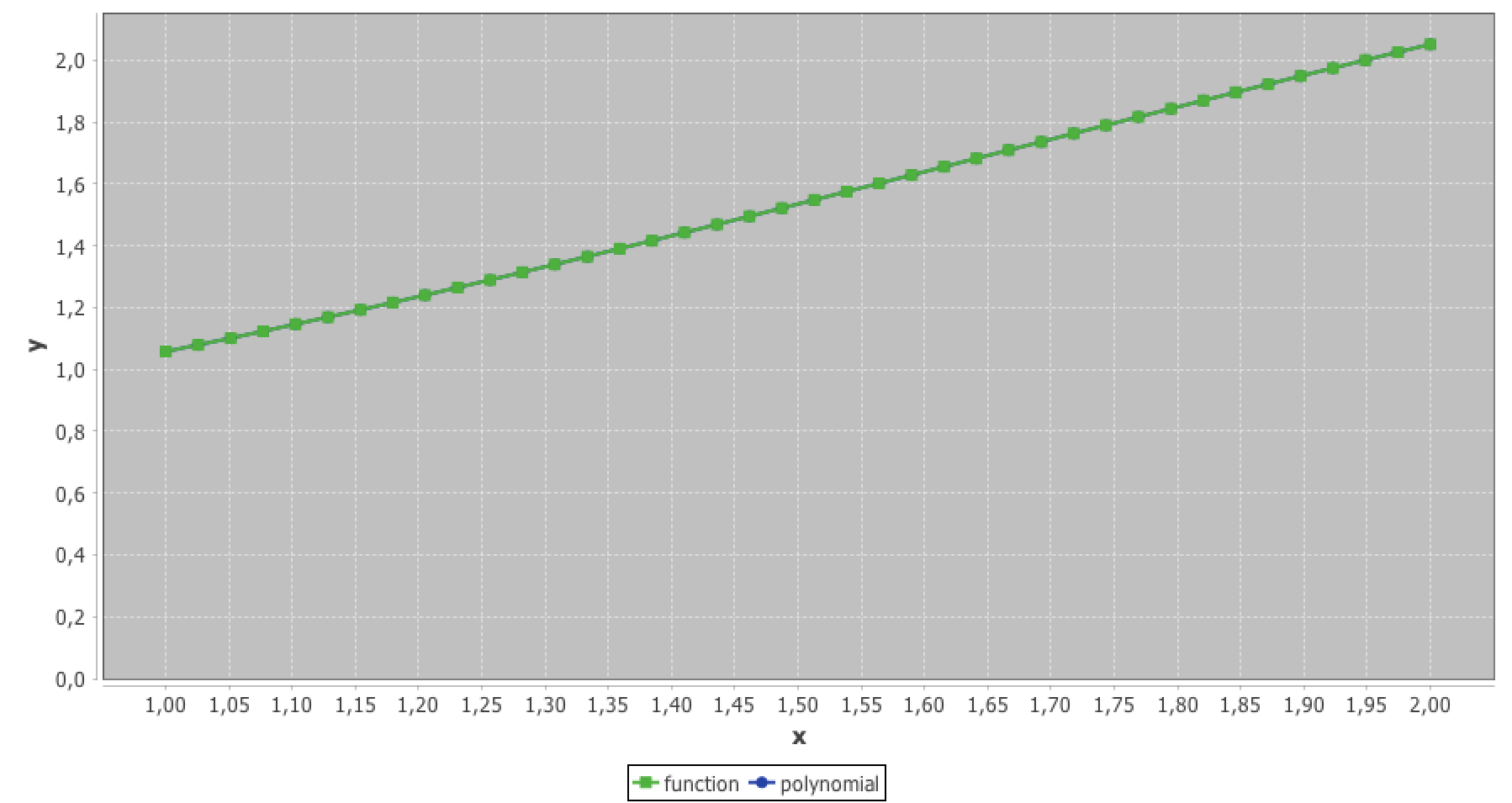
* являются многочленами степени *n*
* при

**Листинг программы**

(Программа реализована на языке Java)

**import** org.jfree.chart.ChartFactory;  
**import** org.jfree.chart.ChartPanel;  
**import** org.jfree.chart.JFreeChart;  
**import** org.jfree.chart.plot.PlotOrientation;  
**import** org.jfree.chart.plot.XYPlot;  
**import** org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;  
**import** org.jfree.data.xy.XYDataset;  
**import** org.jfree.data.xy.XYSeries;  
**import** org.jfree.data.xy.XYSeriesCollection;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
  
**public class** Lagrange **extends** JFrame {  
  
 **private double start** = 1;  
 **private double end** = 2;  
  
  
 **private** Lagrange() **throws** HeadlessException {  
 **super**(**"01 Lagrange"**);  
  
 ChartPanel chartPanel = **new** ChartPanel(createChar());  
  
 **this**.setContentPane(chartPanel);  
  
 **this**.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 **this**.setSize(900, 500);  
 **this**.setVisible(**true**);  
 }  
  
 **private double** function(**double** x) {  
 **return** Math.*pow*(2, -x) + Math.*log*(x + 2) - Math.*cos*(x);  
 }  
  
 **private double** polynomial(**double** x) {  
 **double** res = 0;  
  
 **int** param\_n = 4;  
 **double** step = (**end** - **start**) / param\_n;  
  
 **for** (**int** i = 0; i <= param\_n; i++) {  
 **double** x\_i = **start** + i \* step;  
 **double** sum = function(x\_i);  
  
 **for** (**int** j = 0; j <= param\_n; j++) {  
 **if** (j != i) {  
 **double** x\_j = **start** + j \* step;  
 sum \*= (x - x\_j) / (i - j);  
 }  
 }  
 res += sum;  
 }  
  
 res /= Math.*pow*(step, param\_n);  
  
 **return** res;  
 }  
  
 **private** XYDataset createDataset () {  
 **double** deltaForPlot = 0;  
 **double** plotStart = **start** - deltaForPlot;  
 **double** plotEnd = **end** + deltaForPlot;  
 **int** countOfDotesForPlot = 40;  
  
 **double** step = (**end** - **start**) / (countOfDotesForPlot - 1);  
 XYSeries functionArray = **new** XYSeries(**"function"**);  
 XYSeries polynomialArray = **new** XYSeries(**"polynomial"**);  
 **for** (**double** i = plotStart; i <= plotEnd; i += step) {  
 functionArray.add(i, function(i));  
 polynomialArray.add(i, polynomial(i));  
 }  
  
 XYSeriesCollection dataset = **new** XYSeriesCollection();  
 dataset.addSeries(functionArray);  
 dataset.addSeries(polynomialArray);  
 **return** dataset;  
 }  
  
 **private** JFreeChart createChar() {  
 JFreeChart chart = ChartFactory.*createXYLineChart*(  
 **null**,  
 **"x"**, **"y"**,  
 createDataset(),  
 PlotOrientation.***VERTICAL***,  
 **true**, **false**, **false**);  
  
  
 XYPlot plot = chart.getXYPlot();  
 XYLineAndShapeRenderer renderer = **new** XYLineAndShapeRenderer();  
 **int** FUNCTION = 0;  
 renderer.setSeriesPaint(FUNCTION, **new** Color(78, 176, 61));  
 **int** POLYNOMIAL = 1;  
 renderer.setSeriesPaint(POLYNOMIAL, **new** Color(36, 65, 165));  
 renderer.setSeriesStroke(FUNCTION, **new** BasicStroke(2f));  
 renderer.setSeriesStroke(POLYNOMIAL, **new** BasicStroke(2f));  
  
 plot.setRenderer(renderer);  
  
 **return** chart;  
 }  
  
 **public static void** main(String[] args) {  
 **new** Lagrange();  
 }  
  
}

**Результат**



Для большей наглядности, демонстрация на большем промежутке: