УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Дисциплина: вычислительная математика

Лабораторная работа №4

“Интерполяция функций ”

Выполнил: Тарасов Александр

Группа: P3212

Преподаватель: Перл О.В.

Вариант: Полином Лагранжа

Санкт-Петербург, 2020 г.

Описание метода

Пусть некоторая функция f(x) задана набором точек на некотором интервале [a,b]. Задачей интерполяции является поиск функции F(x), принимающей в точках x_i те же значения y_i.
Точки x_i называют узлами интерполяции.

Существуют различные методы, позволяющие решить задачу интерполяции. Один из них – поиск интерполяционного многочлена Лагранжа.

Суть метода заключается в построении так называемого интерполяционного многочлена Ln(x), который будет являться искомой функцией F(x).

$$Ln(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n$$

$$Ln(x_i) = y_i$$

Ln(x) имеет вид:

$$Ln(x) = l_0(x) + l_1(x) + \cdots + l_n(x)$$

где $l_i(x)$ – полином степени n, который удовлетворяет условию :

$$l_i(x_j) = \begin{cases} y_i, & \text{если } i = j \\ 0, & \text{если } i \neq j \end{cases}$$

Полиномы *l(x)* составляются следующим образом:

$$l_i(x) = c_i(x - x_0)(x - x_1) \dots .(x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)$$

После некоторых преобразований полиномы примут вид:

$$l_0(x) = \frac{(x - x_1)(x - x_2) \dots (x - x_n)}{(x_0 - x_1)(x_0 - x_2) \dots (x_0 - x_n)}$$

$$l_1(x) = \frac{(x - x_0)(x - x_2) \dots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \dots (x_1 - x_n)}$$

$$l_2(x) = \frac{(x - x_0)(x - x_1)(x - x_3) \dots (x - x_n)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3) \dots (x_2 - x_n)}$$

$$l_3(x) = \frac{(x - x_0)(x - x_1)(x - x_2) \dots (x - x_n)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2) \dots (x_3 - x_n)} \dots \dots$$

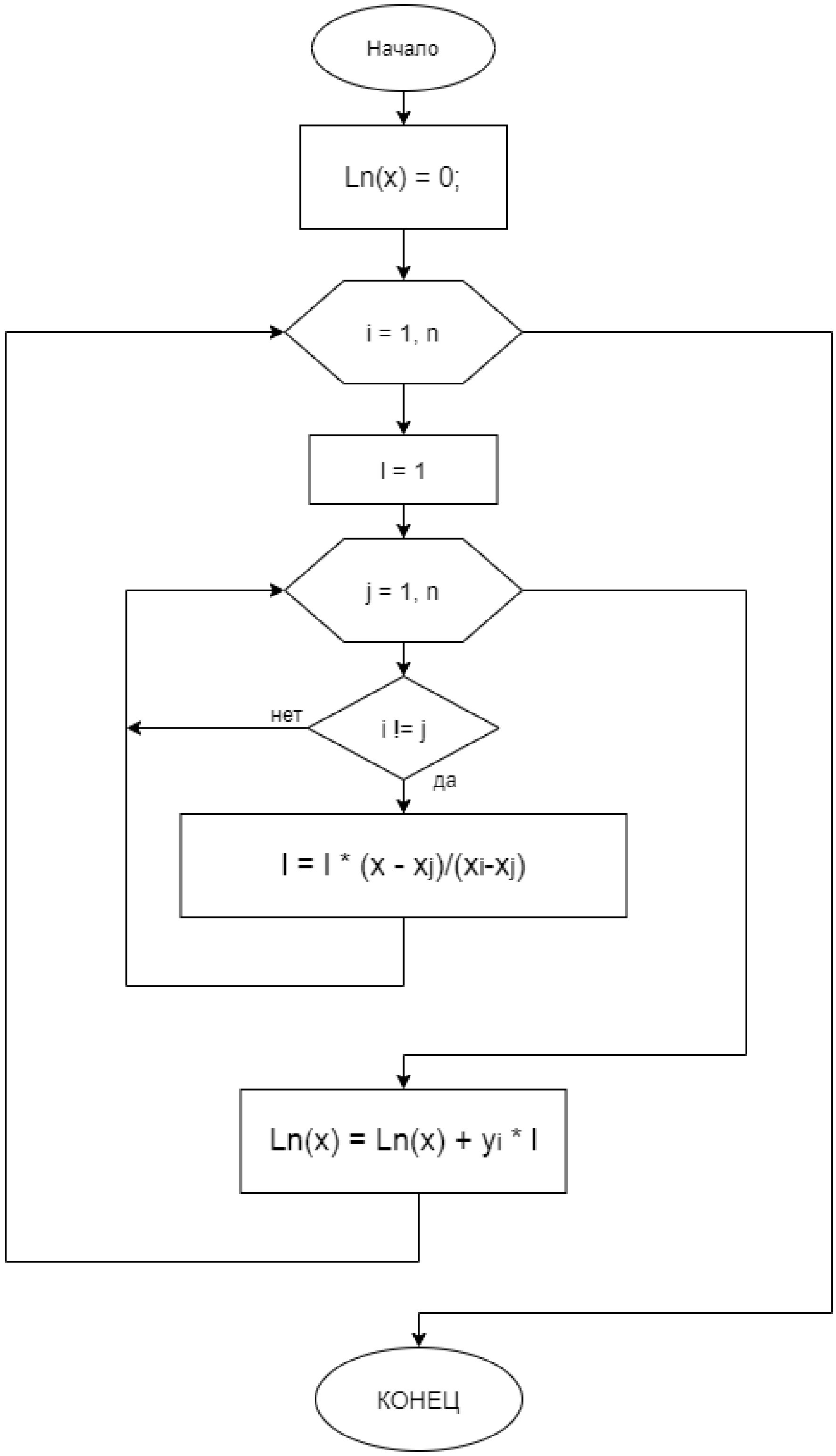
С учетом найденных коэффициентов интерполяционный многочлен Лагранжа запишется в виде:

$$Ln(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

или:

$$Ln(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

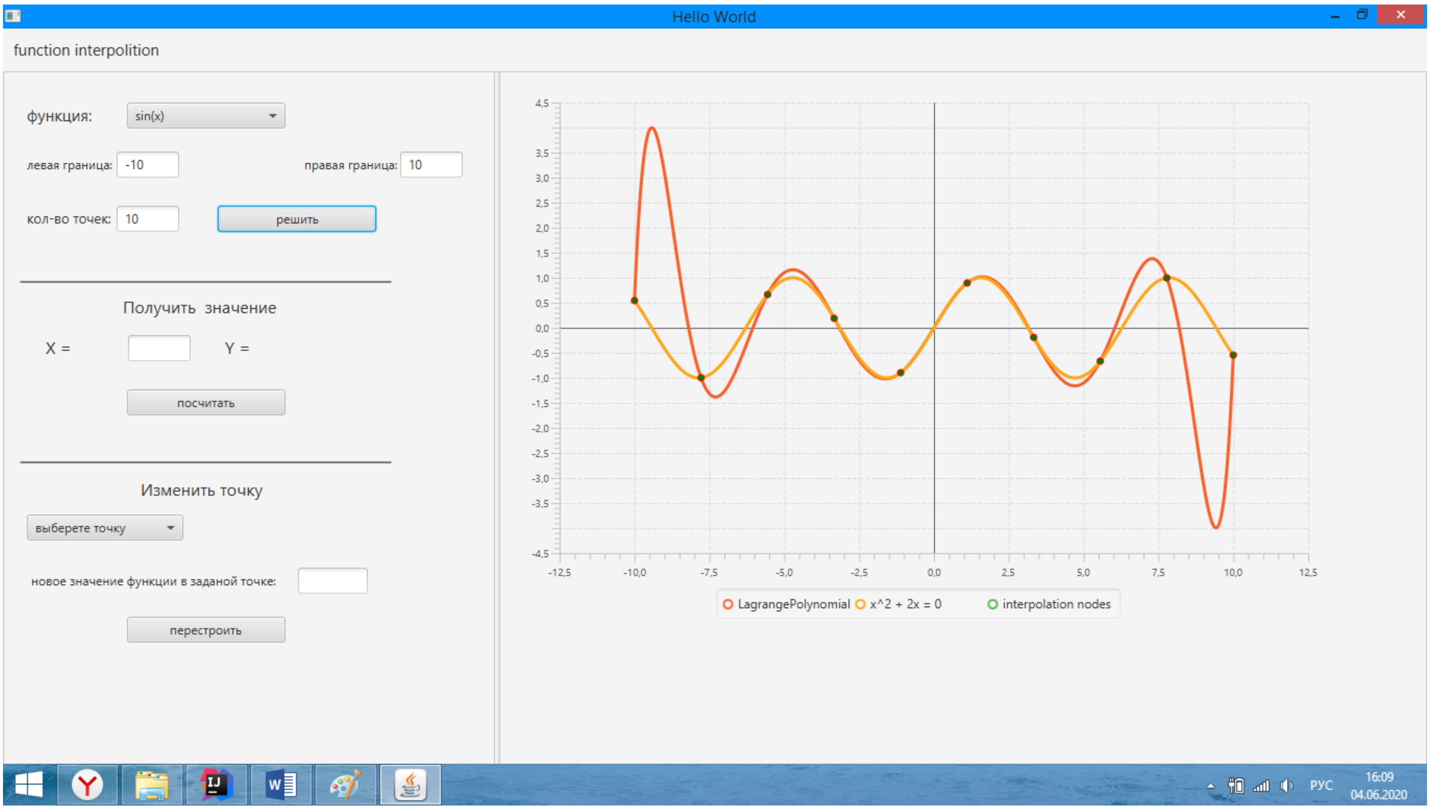
Блок-схема численного метода



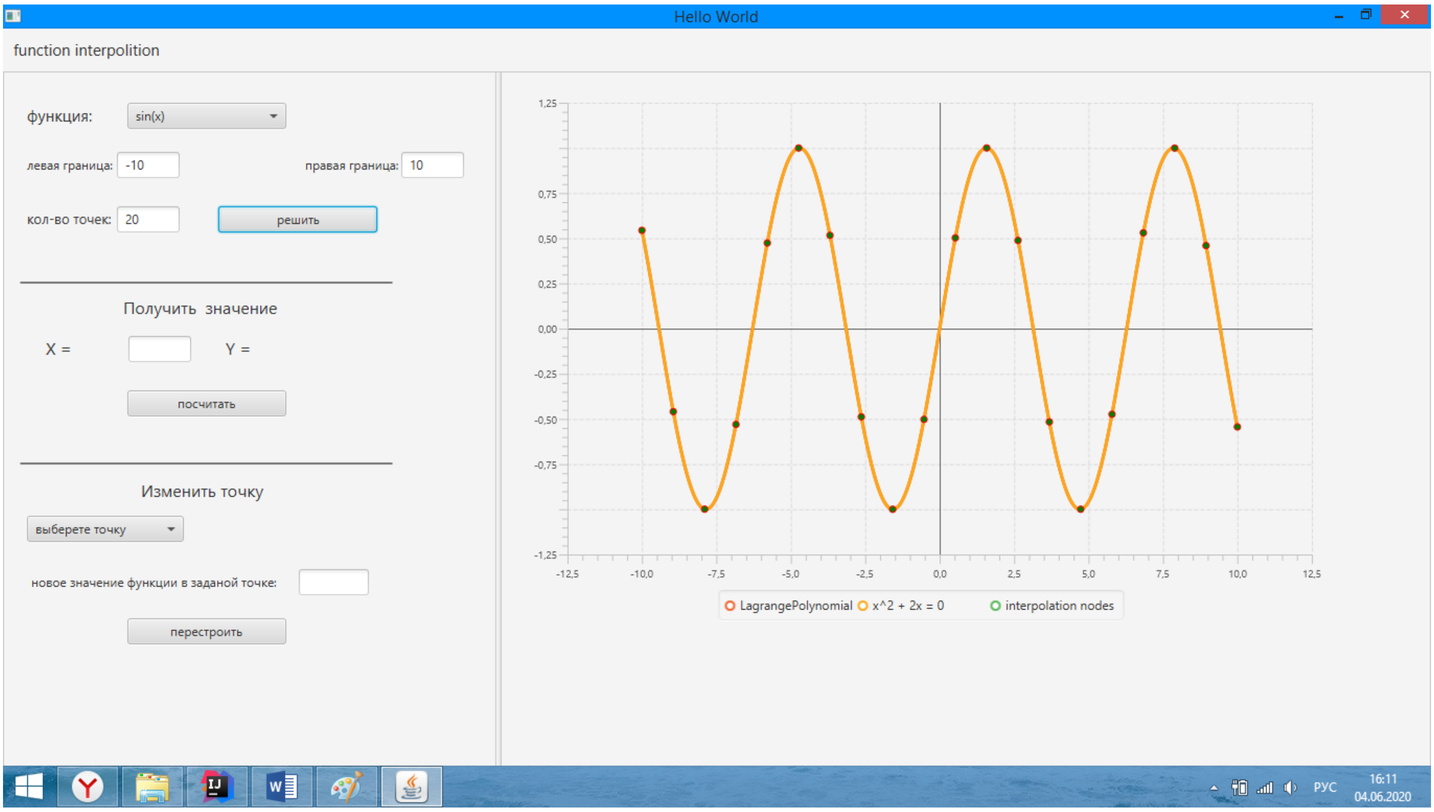
Примеры

f(x) = sin(x);

Кол-во точек 10:



Кол-во точек 10:



Листинг численного метода

LagrangePolynomial.java

```
<…>
public static void initParam(Functions func, double leftBrdr, double rightBrdr, int
dotsCnt){
     function = func;
     leftBorder = leftBrdr;
     rightBorder = rightBrdr;
     dotsCount = dotsCnt;
     xValues = new ArrayList<>();
     yValues = new ArrayList<>();
     prepareData();
 }


 public static void prepareData(){
     double step = Math.abs(rightBorder - leftBorder) / (dotsCount-1);
     double j = leftBorder;

     for (int i = 0; i < dotsCount; i++){
         xValues.add(i, j);
         yValues.add(i, function.solve(j, 0));
         j += step;
     }
 }


 public static void changeValue(int index, double yValue){
     yValues.set(index, yValue);

 }

public static double interpolite (double x)
 {
     double lagrangePol = 0;
     for (int i = 0; i < dotsCount; i++)
     {
         double basicsPol = 1;
         for (int j = 0; j < dotsCount; j++)
         {
             if (j != i)
             {
                 basicsPol *= (x - xValues.get(j))/(xValues.get(i) - xValues.get(j));
             }
         }
         lagrangePol += basicsPol * yValues.get(i);
     }

     return lagrangePol;
 }
<…>
```

AppController.java

```
<…>
@FXML
void solveBtnPressed(ActionEvent event) {
    leftBorder = Double.parseDouble(leftBorderInput.getText());
    rightBorder = Double.parseDouble(rightBorderInput.getText());
    dotsCount = Integer.parseInt(dotsCountInput.getText());

    LagrangePolynomial.initParam(currentFunction, leftBorder, rightBorder,
dotsCount);

    initDotsBox();
    drawGraph();
    drawPoints();

}

@FXML
void resolveBtnClicked(ActionEvent event) {

YOut.setText(String.valueOf(LagrangePolynomial.interpolite(Double.parseDouble(XInput.
getText()))));
}

@FXML
void changeValueBtnCliked(ActionEvent event) {
    LagrangePolynomial.changeValue(Integer.parseInt(dotNumberBox.getValue())-1,
Double.parseDouble(changedYInput.getText()));
    drawGraph();
    drawPoints();
}

private void drawPoints() {
    XYChart.Series<Double, Double> series = new XYChart.Series<>();
    for (int i = 0; i < dotsCount; i++){
        XYChart.Data point = new XYChart.Data(LagrangePolynomial.getxValues().get(i),
LagrangePolynomial.getyValues().get(i));
        point.setNode(new Circle(3.0, Color.GREEN));
        series.getData().add(point);
    }
    series.setName("interpolation nodes");
    chart.getData().add(series);
}

private void drawGraph(){
    double step = 0.02;
    XYChart.Series<Double, Double> series = new XYChart.Series<>();
    series.setName("LagrangePolynomial");
    for (double x = leftBorder; x <= rightBorder + 0.00001; x = x + step) {
        series.getData().add(new XYChart.Data<>(x,
LagrangePolynomial.interpolite(x)));
    }

    XYChart.Series<Double, Double> series2 = new XYChart.Series<>();
    series2.setName(functionsList.get(currentFunctionIndx));
    for (double x = leftBorder; x <= rightBorder + 0.00001; x = x + step) {
        series2.getData().add(new XYChart.Data<>(x, currentFunction.solve(x,0)));
    }
    chart.getData().setAll(series, series2);
} <…>
```

Вывод

Основное различие аппроксимации и интерполяции заключается в условии прохождения полученной функции через заданные точки. В интерполяции в этих точках – узлах интерполяции, полученный полином должен принимать заданные значения. Для аппроксимации данное требование необязательно.

Плюсы интерполяции полиномом Лагранжа:
+ малая погрешность при $n<20$
+ узлы могут быть удалены друг от друга на различное расстояние
+ удобен, когда значения функций меняются, а узлы интерполяции неизменны

Минусы:
- при изменении числа узлов необходимо производить все вычисления заново.

Так же следует отметить метод Ньютона.
+ позволяет не пересчитывать все коэффициенты заново при добавлении новой точки.
- работает только с таблицами с равноудаленными узлами.