Университет ИТМО

Кафедра ВТ

**Вычислительная математика**

Лабораторная работа №3

Интерполирование многочленом Ньютона

Группа P3210

Нгу Фыонг Ань

ПРОВЕРИЛ:

Калёнова Ольга Вячеславовна

2018 год

**Задание:**

Для **интерполяции** необходимо подготовить 3-4 набора данных (в зависимости от функции).

***/\****Исходные данные должны быть подготовлены следующим образом:

* Берем функцию
* Берем точки x (точки не обязательно упорядочены)
* значение y получаем на основе данных выбранной функции

**Например**:

* берем sinx
* 1) берем 3-4 точки на интервале 0 по 2Пи(шаг более менее большой)  
  2) берем 8-10 точек на интервале 0 по 2Пи (уменьшаем шаг)  
  3) точки с предыдущего примера, только для одной точки изменяем значение y, например  
  было 0.8, делаем -5, смотрим как ведет себя интерполяция.  
  4) берем 8-10 точек на интервале 0 по 50Пи

***\*/***

В итоге, должны получить график, на котором одним цветом исходная функция (sinx), а другим цветом полученный график в результате интерполяции, и на графике должны быть отмечены сами **точки** (узлы) интерполяции.

Интерполяционный график должен пройти через исходные эти точки.

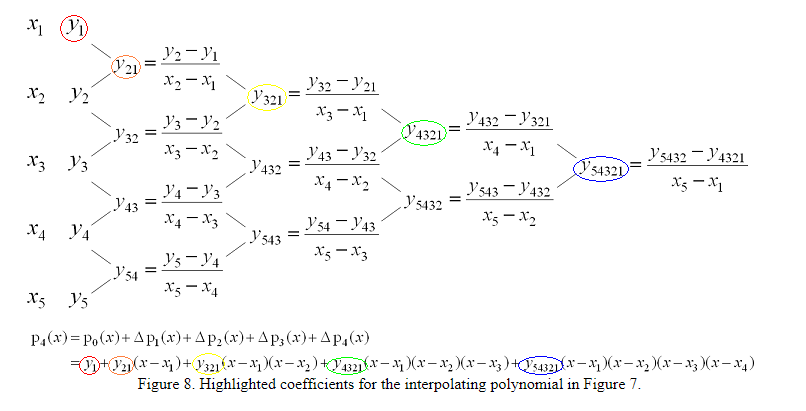
Программа должна позволять найти значение y (отдельное поле) для любого введенного x

(рассчитывается на основе построенного интерполяционного многочлена).

**Описание метода**

The interpolating Newton polynomial of degree *n* − 1 is

p*n*− 1(*x*) = *y*1 + *y*21(*x* − *x*1) + *y*321(*x* − *x*1)(*x* − *x*2) + ··· *yn*···321(*x* − *x*1)(*x* − *x*2)···(*x* − *xn*− 1)



**Код программы:**

**#Newton**

public class Newton extends Application {

public static void main(String[] args) {

launch(args);

}

@Override

public void start(final Stage stage) {

double[] x = new double[20];

double[] y = new double[20];

double[] f = new double[20];

int code;

System.out.println("(1) y = sin(x)");

System.out.println("(2) x^2 + 3\*x - 4");

System.out.println("Please choose your equation (1/2)");

Scanner sc = new Scanner(System.in);

String input;

while (true){

input = sc.nextLine();

if (input.equals("1")) {

code = 1;

break;

} else if (input.equals("2")) {

code = 2;

break;

} else {

System.out.println("Invalid code. Please choose your equation (1/2)");

}

}

System.out.println("Please enter x, when you are finished, enter \*ok\* :");

int v =0;

double u;

while (true){

input = sc.nextLine();

if (input.equals("ok")) {

break;

}

try {

u = Double.parseDouble(input);

x[v] = u;

v++;

} catch (NumberFormatException e){

System.out.println(input + " is not a valid value. Please continue to enter x");

}

}

int n = v;

if (code ==1) {

for (int i=0; i<n; i++) y[i] = Math.sin(x[i]);

} else {

for (int i=0; i<n; i++) y[i] = x[i]\*x[i] + 3\*x[i] -4;

}

double [][] s = new double [20][20];

for (int i = 0; i < n; i++) {

s[i][0] = x[i];

s[i][1] = y[i];

}

int dis = 0;

for (int j = 2; j<= n+1; j++){

dis++;

int start = 0;

for (int i = 0; i<n-dis; i++){

s[i][j] = (s[i+1][j-1]-s[i][j-1])/(s[start+dis][0]-s[start][0]);

start++;

}

}

for (int i=0; i<n; i++) f[i] = s[0][i+1];

double[] result = new PolynomProduct(x,f).getresult();

Axes axes = new Axes(

800, 600,

-10, 10, 1,

-10, 10, 1

);

Plot plot = new Plot(

z -> {

double sum = 0;

for (int i = 0; i < n; i++) {

sum = sum + result[i] \* Math.pow(z, i);

}

return sum;

},

-10, 10, 0.1,

axes, Color.BLUE);

for (int i = 0; i < n; i++) {

plot.getChildren().addAll(new Circle(400 + 40\*x[i], 300 - 30 \* y[i], 3, Color.BLACK));

}

if (code == 1) {

Plot plot1 = new Plot(

z -> Math.sin(z),

-10, 10, 0.1,

axes, Color.RED

);

StackPane layout = new StackPane(plot, plot1);

layout.setPadding(new Insets(20));

layout.setStyle("-fx-background-color: rgb(255,255,255);");

stage.setTitle("Graphic");

stage.setScene(new Scene(layout, Color.WHITE));

stage.show();

}

if (code == 2) {

Plot plot1 = new Plot(

z -> z \* z + 3 \* z - 4,

-6, 6, 0.1,

axes, Color.RED

);

StackPane layout = new StackPane(plot, plot1);

layout.setPadding(new Insets(20));

layout.setStyle("-fx-background-color: rgb(35, 39, 50);");

stage.setTitle("Graphic");

stage.setScene(new Scene(layout, Color.WHITE));

stage.show();

}

}

class Axes extends Pane {

private NumberAxis xAxis;

private NumberAxis yAxis;

public Axes(

int width, int height,

double xLow, double xHi, double xTickUnit,

double yLow, double yHi, double yTickUnit

) {

setMinSize(Pane.USE\_PREF\_SIZE, Pane.USE\_PREF\_SIZE);

setPrefSize(width, height);

setMaxSize(Pane.USE\_PREF\_SIZE, Pane.USE\_PREF\_SIZE);

xAxis = new NumberAxis(xLow, xHi, xTickUnit);

xAxis.setSide(Side.BOTTOM);

xAxis.setMinorTickVisible(false);

xAxis.setPrefWidth(width);

xAxis.setLayoutY(height / 2);

yAxis = new NumberAxis(yLow, yHi, yTickUnit);

yAxis.setSide(Side.LEFT);

yAxis.setMinorTickVisible(false);

yAxis.setPrefHeight(height);

yAxis.layoutXProperty().bind(

Bindings.subtract(

(width / 2) + 1,

yAxis.widthProperty()

)

);

getChildren().setAll(xAxis, yAxis);

}

public NumberAxis getXAxis() {

return xAxis;

}

public NumberAxis getYAxis() {

return yAxis;

}

}

class Plot extends Pane {

public Plot(

Function<Double, Double> f,

double xMin, double xMax, double xInc,

Axes axes, Color color

) {

Path path = new Path();

path.setStroke(color);

path.setStrokeWidth(2);

path.setClip(

new Rectangle(

0, 0,

axes.getPrefWidth(),

axes.getPrefHeight()

)

);

double x = xMin;

double y = f.apply(x);

path.getElements().add(

new MoveTo(

mapX(x, axes), mapY(y, axes)

)

);

x += xInc;

while (x < xMax) {

y = f.apply(x);

path.getElements().add(

new LineTo(

mapX(x, axes), mapY(y, axes)

)

);

x += xInc;

}

setMinSize(Pane.USE\_PREF\_SIZE, Pane.USE\_PREF\_SIZE);

setPrefSize(axes.getPrefWidth(), axes.getPrefHeight());

setMaxSize(Pane.USE\_PREF\_SIZE, Pane.USE\_PREF\_SIZE);

getChildren().setAll(axes, path);

}

private double mapX(double x, Axes axes) {

double tx = axes.getPrefWidth() / 2;

double sx = axes.getPrefWidth() /

(axes.getXAxis().getUpperBound() -

axes.getXAxis().getLowerBound());

return x \* sx + tx;

}

private double mapY(double y, Axes axes) {

double ty = axes.getPrefHeight() / 2;

double sy = axes.getPrefHeight() /

(axes.getYAxis().getUpperBound() -

axes.getYAxis().getLowerBound());

return -y \* sy + ty;

}

}

}

**#** **PolynomProduct**

public class PolynomProduct {

double [] x, f;

public PolynomProduct(double[] x, double[] f ) {

this.x = x;

this.f = f;

}

public double[] getresult(){

double[] result = new double[x.length];

for (int i = 0; i < x.length; ++i) {

result[i] = 0.0;

}

for (int i = 0; i < x.length; ++i) {

double[] poly = getPoly(x, i - 1);

for (int j = 0; j < poly.length; ++j) {

result[j] += poly[j] \* f[i];

}

}

return result;

}

public static double[] getPoly(double[] x, int i) {

double[] coefs = {1.0};

for (int j = 0; j <= i; ++j) {

double[] coefsLocal = {-x[j], 1.0};

coefs = getPolyProduct(coefs, coefsLocal);

}

return coefs;

}

public static double[] getPolyProduct(double[] coefs1, double[] coefs2) {

int s1 = coefs1.length - 1;

int s2 = coefs2.length - 1;

int degree = s1 + s2;

double[] coefsProduct = new double[degree + 1];

for (int k = 0; k <= degree; ++k) {

coefsProduct[k] = 0.0;

}

for (int i = 0; i <= s1; ++i) {

for (int j = 0; j <= s2; ++j) {

coefsProduct[i + j] += coefs1[i] \* coefs2[j];

}

}

return coefsProduct;

}

}

**Тест:**

**Ввод программы:**

(1) y = sin(x)

(2) x^2 + 3\*x - 4

Please choose your equation (1/2)

1

Please enter x, when you are finished, enter \*ok\* :

1

3

7

9

-2

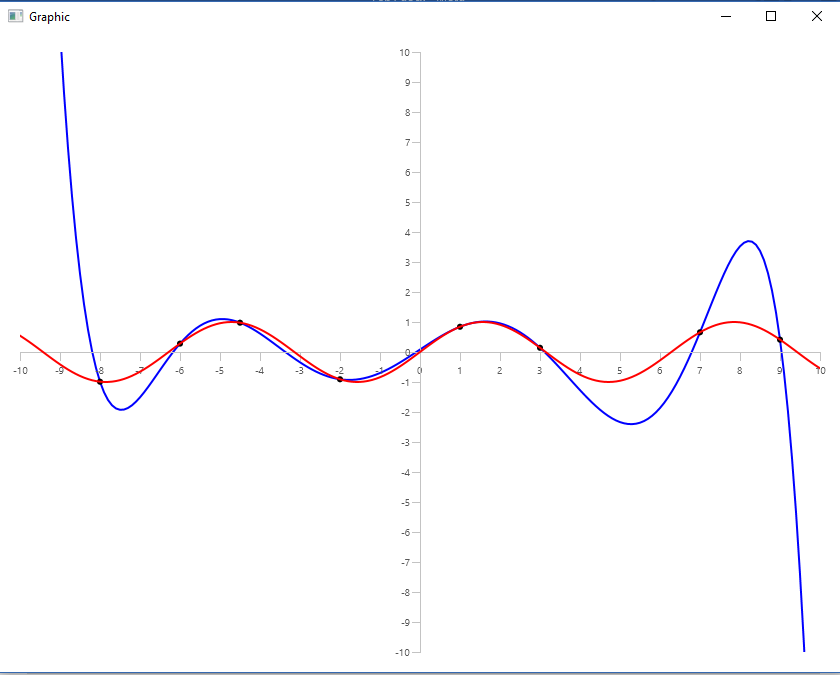
-6

-4.5

-8

ok

**Вывод программы:**



**Вывод:**