БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра вычислительной математики

Лабораторная работа № 2

**Приближение функции с помощью кубического сплайна**

**Студент**: Шелег Владислава

2 курс, 9 группа

**Преподаватель**: Никифоров И. В

МИНСК

2017

1. **Постановка задачи**

Пусть дана функция . Необходимо построить кубический сплайн, интерполирующий заданную функцию.

1. **Алгоритм решения**

На каждом отрезке функция S(x) есть полином 3 степени , коэффициенты которого надо определить.

(x - ) +

Тогда , ,

Условия непрерывности всех производных до второго порядка включительно:

, ,

Условия интерполяции:

Пусть = - ,

Тогда коэффициенты сплайна можно вычислить по формулам:

*=* 6

= - +

**3.Листинг**

**public class** Spline **extends** JFrame {

**private double** function(**double** x) {  
 **return** Math.pow(2, x) - Math.log(x + 2) + Math.cos(x);  
 }

**private void** buildSpline() {  
 splines = **new** SplineTuple[param\_n];  
 **double**[] x = **new double**[param\_n];  
 **double**[] y = **new double**[param\_n];  
  
 **double** step = (end - begin) / (param\_n - 1);  
  
 **for** (**int** i = 0; i < param\_n; ++i) {  
 **double** x\_i = begin + i \* step;  
 x[i] = x\_i;  
 y[i] = function(x\_i);  
 splines[i] = **new** SplineTuple(x[i], y[i]);  
 }  
 splines[0].c = splines[param\_n - 1].c = 0.0;  
 **double**[] alpha = **new double**[param\_n - 1];  
 **double**[] beta = **new double**[param\_n - 1];  
 alpha[0] = beta[0] = 0.0;  
 **for** (**int** i = 1; i < param\_n - 1; ++i) {  
 **double** A = x[i] - x[i - 1];  
 **double** B = x[i + 1] - x[i];  
 **double** C = 2.0 \* (A + B);  
 **double** F = 6.0 \* ((y[i + 1] - y[i]) / B - (y[i] - y[i - 1]) / A);  
 **double** z = (A \* alpha[i - 1] + C);  
 alpha[i] = -B / z;  
 beta[i] = (F - A \* beta[i - 1]) / z;  
 }  
 **for** (**int** i = param\_n - 2; i > 0; --i) {  
 splines[i].c = alpha[i] \* splines[i + 1].c + beta[i];  
 }  
 **for** (**int** i = param\_n - 1; i > 0; --i) {  
 **double** hi = x[i] - x[i - 1];  
 splines[i].d = (splines[i].c - splines[i - 1].c) / hi;  
 splines[i].b = hi \* (2.0 \* splines[i].c + splines[i - 1].c) / 6.0 + (y[i] - y[i - 1]) / hi;  
 }  
 }

**private double** interpolate(**double** x) {  
 **if** (splines == **null**) {  
 **return** 0;  
 }  
 **int** n = splines.length;  
 SplineTuple s;  
  
 **if** (x <= splines[0].x) {  
 s = splines[0];  
 } **else if** (x >= splines[n - 1].x) {  
 s = splines[n - 1];  
 } **else** {  
 **int** i = 0;  
 **int** j = n - 1;  
 **while** (i + 1 < j) {  
 **int** k = i + (j - i) / 2;  
 **if** (x <= splines[k].x) {  
 j = k;  
 } **else** {  
 i = k;  
 }  
 }  
 s = splines[j];  
 }  
  
 **double** dx = x - s.x;  
 **return** s.a + (s.b + (s.c / 2.0 + s.d \* dx / 6.0) \* dx) \* dx;  
 }  
}

**4.Результат**

