**Отчет**

**по Методам численного анализа**

Исполнитель

студент 161 группы

специальности

Прикладная математика

Борис Д. Ю.

«4» Мая 2018 г.

Гродно, 2018

**ЛАБОРАТОРНАЯ РАБОТА 8**

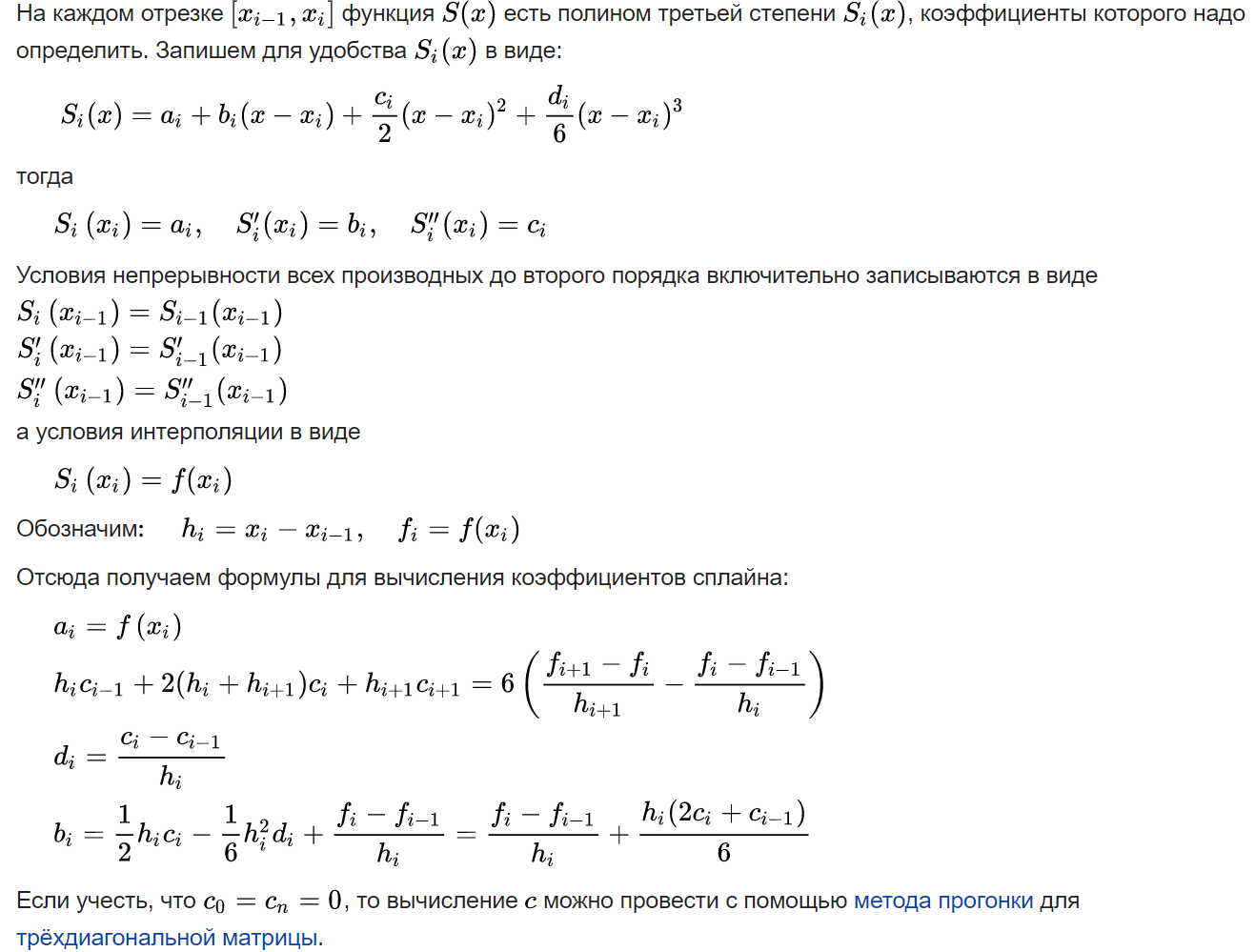
**Тема. Интерполирование сплайнами**

Построить график сплайна (параболический) и отметить на нем узлы интерполяции. Вычислить значения сплайна в некоторых точках.

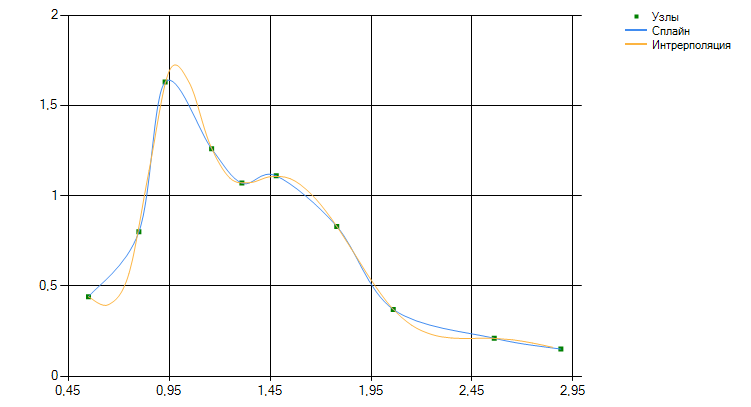
Условие:



1. **Теоретический материал**



1. **Результат вычислений**



1. **Текст программы**

class spline

{

SplineTuple[] splines; // Сплайн

// Структура, описывающая сплайн на каждом сегменте сетки

private struct SplineTuple

{

public double a, b, c, d, x;

}

// Построение сплайна

// x - узлы сетки, должны быть упорядочены по возрастанию, кратные узлы запрещены

// y - значения функции в узлах сетки

// n - количество узлов сетки

public void BuildSpline(double[] x, double[] y, int n)

{

// Инициализация массива сплайнов

splines = new SplineTuple[n];

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

{

splines[i].x = x[i];

splines[i].a = y[i];

}

splines[0].c = splines[n - 1].c = 0.0;

// Решение СЛАУ относительно коэффициентов сплайнов c[i] методом прогонки для трехдиагональных матриц

// Вычисление прогоночных коэффициентов - прямой ход метода прогонки

double[] alpha = new double[n - 1];

double[] beta = new double[n - 1];

alpha[0] = beta[0] = 0.0;

for ( int i = 1 ; i < n - 1 ; ++i )

{

double hi = x[i] - x[i - 1];

double hi1 = x[i + 1] - x[i];

double A = hi;

double C = 2.0 \* ( hi + hi1 );

double B = hi1;

double F = 6.0 \* ( ( y[i + 1] - y[i] ) / hi1 - ( y[i] - y[i - 1] ) / hi );

double z = ( A \* alpha[i - 1] + C );

alpha[i] = -B / z;

beta[i] = ( F - A \* beta[i - 1] ) / z;

}

// Нахождение решения - обратный ход метода прогонки

for ( int i = n - 2 ; i > 0 ; --i )

{

splines[i].c = alpha[i] \* splines[i + 1].c + beta[i];

}

// По известным коэффициентам c[i] находим значения b[i] и d[i]

for ( int i = 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;

}

}

// Вычисление значения интерполированной функции в произвольной точке

public double Interpolate(double x)

{

if ( splines == null )

{

return double.NaN; // Если сплайны ещё не построены - возвращаем NaN

}

int n = splines.Length;

SplineTuple s;

if ( x <= splines[0].x ) // Если x меньше точки сетки x[0] - пользуемся первым эл-тов массива

{

s = splines[0];

}

else if ( x >= splines[n - 1].x ) // Если x больше точки сетки x[n - 1] - пользуемся последним эл-том массива

{

s = splines[n - 1];

}

else // Иначе x лежит между граничными точками сетки - производим бинарный поиск нужного эл-та массива

{

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;

}

}