МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**

**ОТЧЁТ**

**по лабораторной работе №4**

**по дисциплине: «** *Вычислительная математика* **»**

**Вариант 2**

Выполнил(а):Проверил:

Студенты гр. *АП-227* *Ландовский В.В.*

*Бузмаков А.И.*

*Шестаков К.Д.*

*Федотов И.В.*

«\_\_\_» \_\_\_\_\_\_ 20\_\_г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2024

**Цель работы**

Знакомство с некоторыми методами аппроксимации функций. Получение практических навыков разработки алгоритмов и программной реализации данных методов.

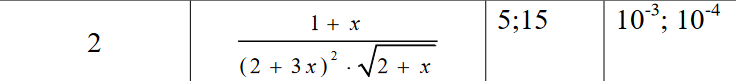
**Постановка задачи**

1. Проанализировать поведение подынтегральной функции, заданной согласно варианту задания в таблице 5.2. Вычислить оценки шаг интегрирования для заданных преподавателем методов и значений точности из таблицы 5.2 используя формулы (5.2), (5.5), (5.7), (5.9). Расчеты выполнить вручную или с использованием любых доступных математических программных пакетов.

2. Разработать программную реализацию вычисления интеграла заданными методами с контролем точности основанном на правиле Рунге с учетом порядка точности метода. Входной информацией для разработанной программы должны быть: пределы интегрирования и требуемая точность Подынтегральная функция жестко задается в программе. Выходная информация: значение интеграла и шаг, при котором оно вычислено. При разработке алгоритма стараться по возможности минимизировать вычислительные затраты. Ограничений на среду разработки не накладывается.

3. Сравнить оценки шага со значениями шага, которые получены в результате работы программ, а значения интеграла полученные программно с результатом, полученным в стороннем математическом программном пакете с высокой точностью.

**Исходные данные**

****

1. Метод правых прямоугольников,
2. Метод средних прямоугольников,
3. 3 степени.

**Ход работы**

Сначала вручную посчитали коэффициенты кубического сплайна и построили его график.

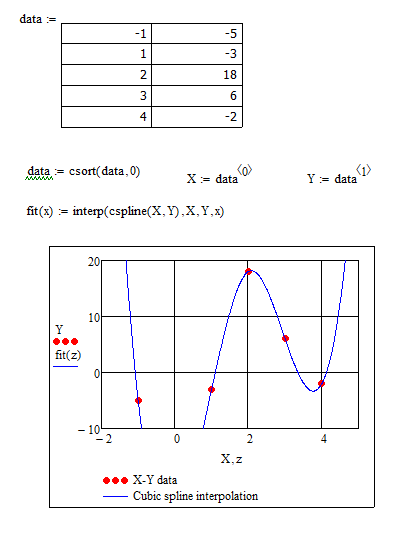


Рис. 1 – График кубического сплайна

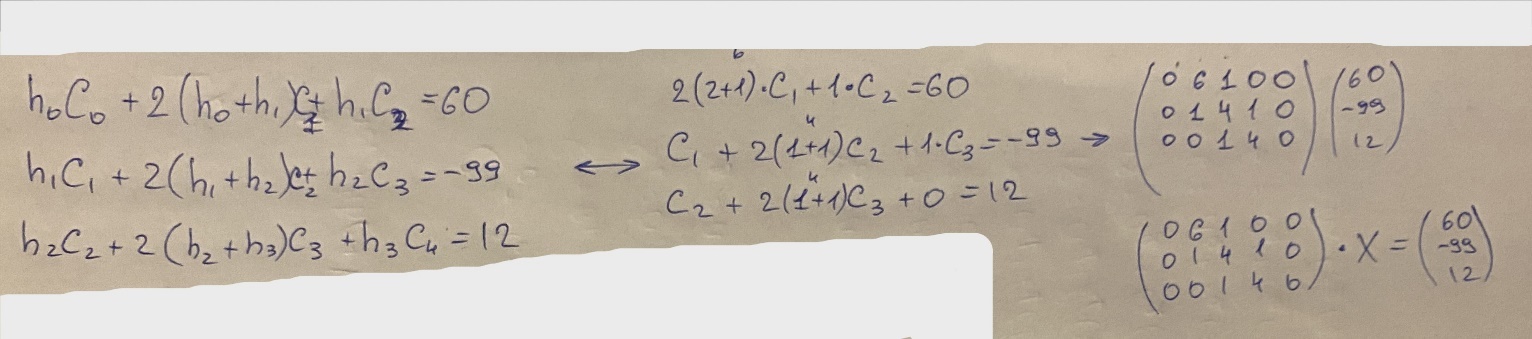


Рис. 2.1 – Расчет коэффициентов “c”

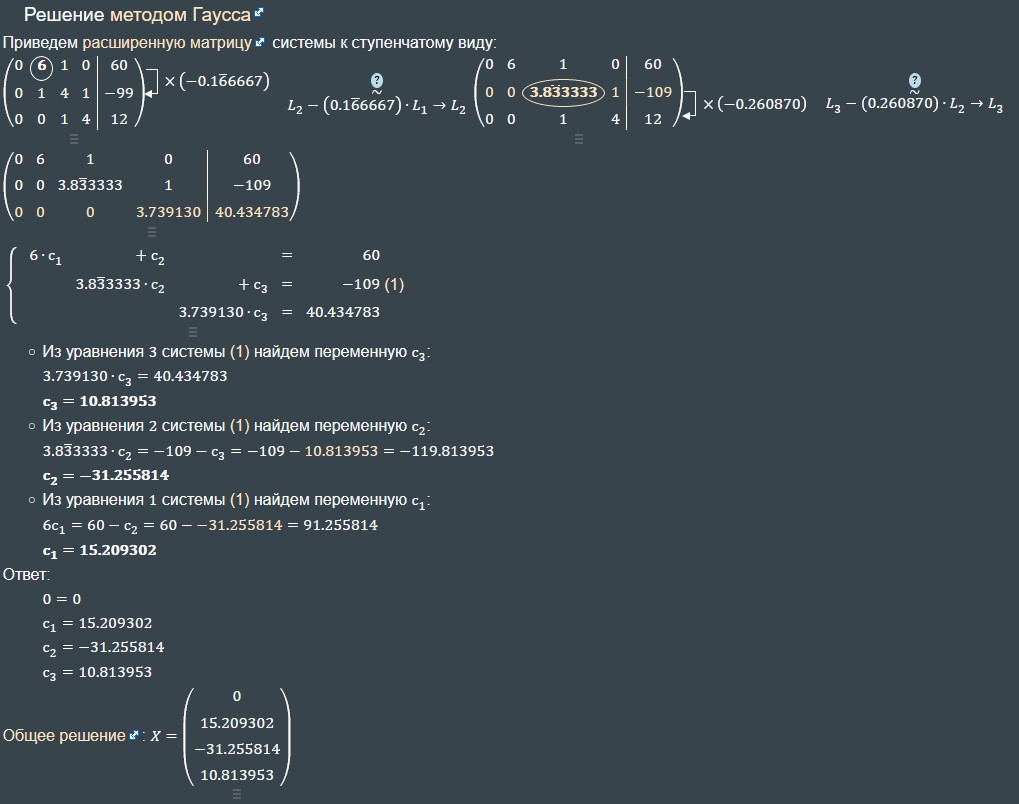


Рис. 2.2 – Расчет коэффициентов “c”

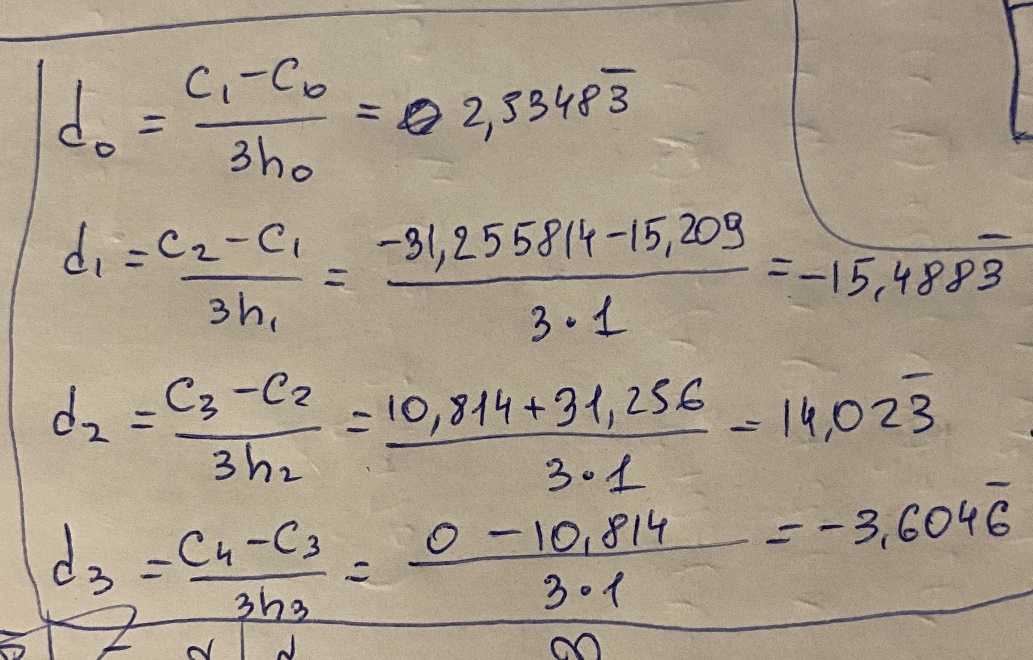


Рис. 3 - Расчет коэффициентов “d”

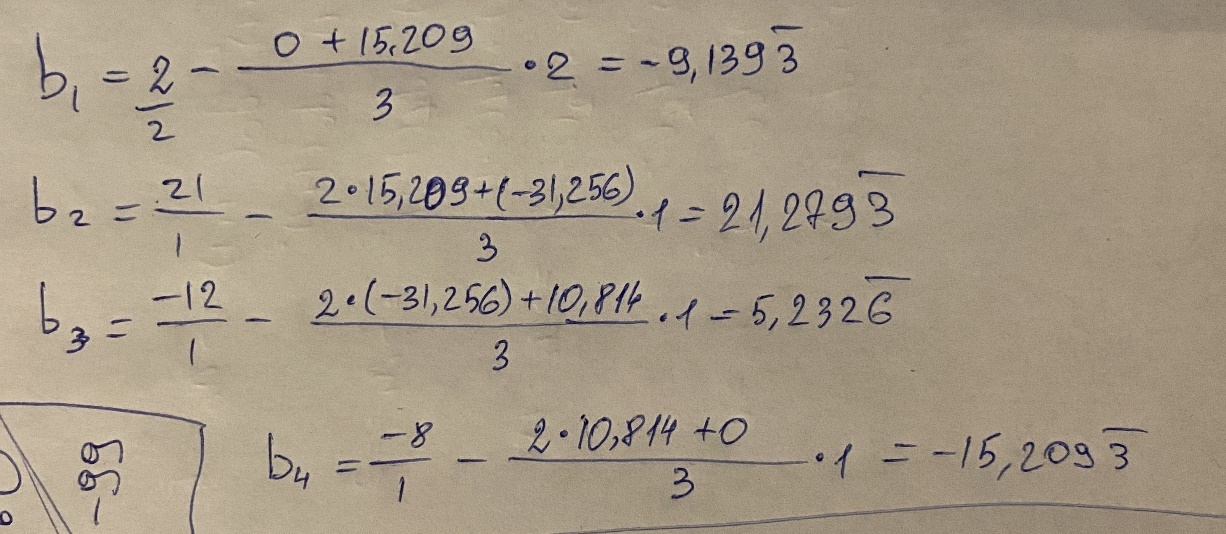


Рис. 4 - Расчет коэффициентов “b”

Затем на основе проведенных нами вычислений реализовали программу.

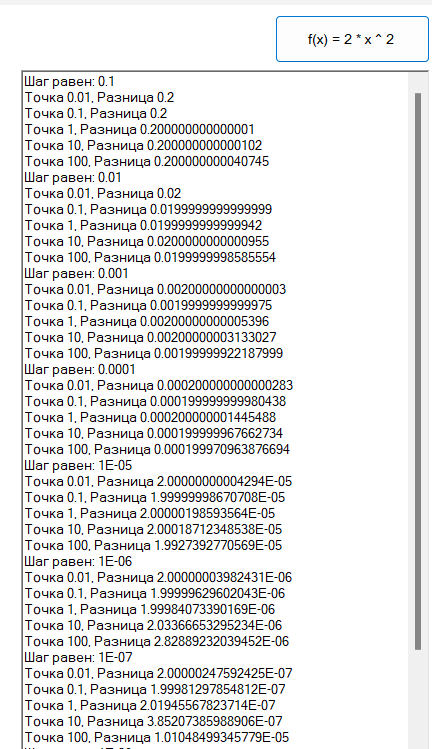


Рис. 5 – Вычисление первой производной f(x) = 2 \* x ^ 2

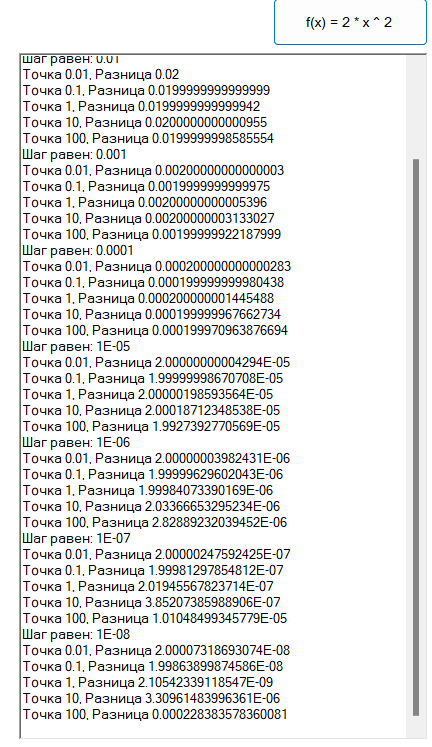


Рис. 5.1 - Вычисление первой производной f(x) = 2 \* x ^ 2

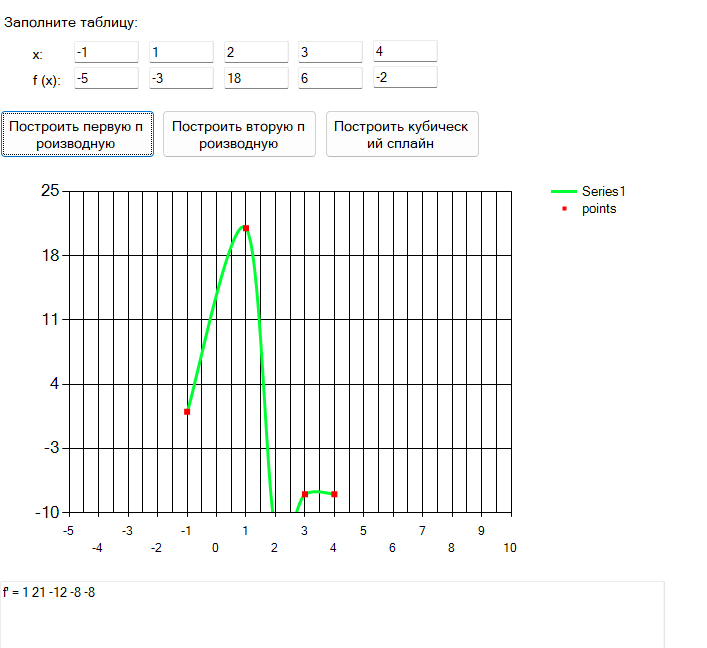


Рис. 6 – График первой производной

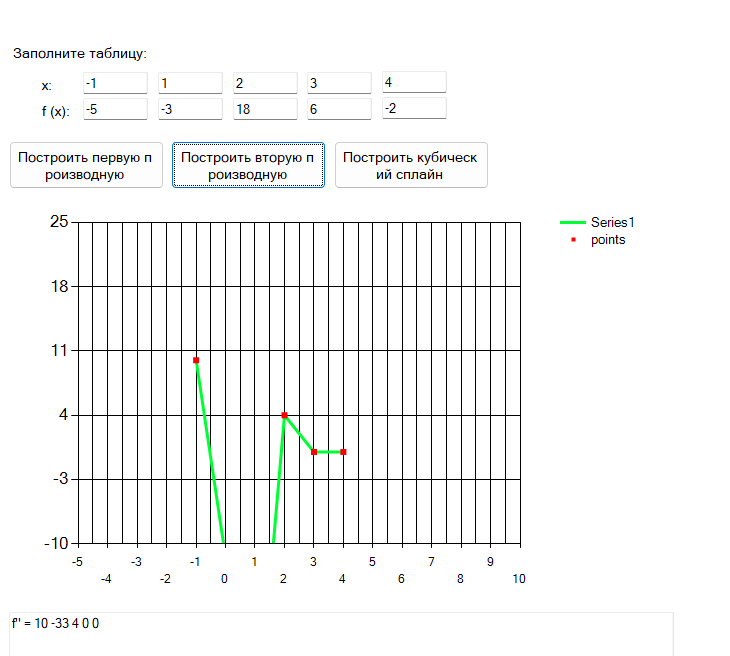


Рис. 7 – График второй производной

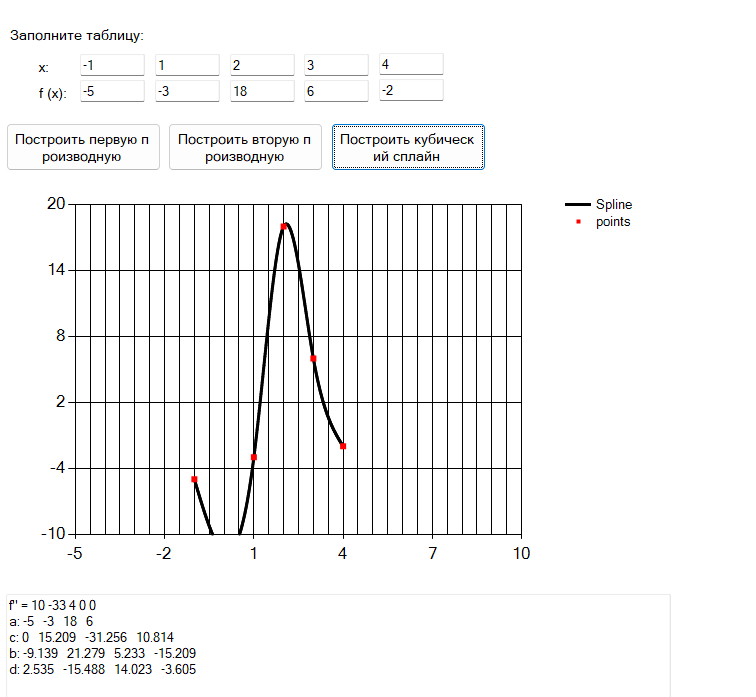


Рис. 8 – График кубического сплайна

**Объяснение полученных результатов**

В данной работе были рассмотрены методы аппроксимации функций с целью приближенного вычисления производных и построения кубического сплайна.

1. **Приближенные вычисления первой производной**:
   * В ходе экспериментов были выполнены вычисления первой производной функции f(x)=2x2 в точках x=10−2,10−1,1,10,102 с различными значениями шага от 10−1 до 10−8.
   * Экспериментально был подобран оптимальный шаг, обеспечивающий наилучшую точность при вычислении производной.
   * Результаты экспериментов показали, что при уменьшении шага точность вычислений увеличивается, однако при слишком маленьких значениях возникают численные ошибки из-за ограниченной точности представления вещественных чисел.
2. **Расчет кубического сплайна вручную**:
   * По заданной таблице значений функции были расчитаны коэффициенты кубического сплайна вручную согласно указанным формулам.
   * Построены графики полученных сплайнов.
3. **Разработка программной реализации кубического сплайна**:
   * На основе заданных формул была разработана программа для построения кубического сплайна для произвольной таблицы значений функции.
   * Также были рассчитаны и построены графики первой и второй производной приближенной функции с использованием выражения для первой производной ​.

**Код разработнной программы**

|  |
| --- |
| Form1.cs |
| using System;  using System.Drawing;  using System.Windows.Forms;  using System.Windows.Forms.DataVisualization.Charting;  namespace vchmat4  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  int n;  TextBox[] X = new TextBox[7]; //массив ячеек x  TextBox[] F = new TextBox[7]; //массив ячеек f  double step = 0.01;  //создание массива с данными из текстбокса  private double[] array(TextBox[] box, int n)  {  double[] x = new double[n];  //заполняем массив x из таблицы  for (int i = 0; i < n; i++)  {  x[i] = new double();  x[i] = double.Parse(box[i].Text);  }  return x;  }  //cтирает данные  private void cleaning(int n)  {  chart1.Series.Add("points");  for (int i = 0; i < n; i++)  {  X[i].Clear();  F[i].Clear();  }  chart1.Series[0].Points.Clear();  chart1.Series[1].Points.Clear();  }  //проверка на пустоту  private bool empty(TextBox box)  {  if (box.Text == "")  return true;  else  return false;  }  //проверка  private bool checkTable()  {  int fl1 = 0;  //проверка на пустоту  for (int i = 0; i < n; i++)  if (empty(X[i]) || empty(F[i]))  {  MessageBox.Show("Заполните таблицу.");  return false;  }  //проверяем, что не все х=0  for (int i = 0; i < n; i++)  if (double.Parse(X[i].Text) == 0)  fl1++;  if (fl1 == n)  {  MessageBox.Show("Невозможно построить график. Введите другие значения х.");  cleaning(n);  return false;  }  //проверка, чтобы пользователь вводил различные точки  for (int i = 0; i < n - 1; i++)  for (int j = i + 1; j < n; j++)  if ((double.Parse(X[i].Text) == double.Parse(X[j].Text)) && (double.Parse(F[i].Text) == double.Parse(F[j].Text)))  {  MessageBox.Show("Невозможно построить график. Введите различные точки.");  cleaning(n);  return false;  }    return true;  }  //кнопка ввести  //пользователь вводит размер таблицы, делает лишние ячейки(максимум ячеек установим 7) невидимыми  private void button1\_Click(object sender, EventArgs e)  {  if(empty(textBox1))  {  MessageBox.Show("Укажите количество аргументов.");  return;  }  n = int.Parse(textBox1.Text);  if ((int.Parse(textBox1.Text) < 1) || (int.Parse(textBox1.Text) > 7))  {  MessageBox.Show("Невозможно построить график. Введите n от 1 до 7.");  textBox1.Clear();  return;  }  X[0] = textBox2;  X[1] = textBox3;  X[2] = textBox5;  X[3] = textBox4;  X[4] = textBox7;  X[5] = textBox6;  X[6] = textBox8;  F[0] = textBox15;  F[1] = textBox14;  F[2] = textBox13;  F[3] = textBox12;  F[4] = textBox11;  F[5] = textBox10;  F[6] = textBox9;  for (int i = 0; i < 7; i++)  {  if (i < n)  {  X[i].Visible = true;  F[i].Visible = true;  }  else  {  X[i].Visible = false;  F[i].Visible = false;  }  X[i].Clear();  F[i].Clear();  }  cleaning(n);  label1.Visible = false;  button1.Visible = false;  textBox1.Visible = false;  }  //построение графика, настраиваются оси  private void graph(double[] x, double[] y)  {  chart1.Series[1].Points.Clear();  chart1.Series[0].Color = Color.FromArgb(0, 255, 50);  chart1.ChartAreas[0].AxisX.Minimum = -5;  chart1.ChartAreas[0].AxisX.Maximum = 10;  chart1.ChartAreas[0].AxisY.Minimum = -10;  chart1.ChartAreas[0].AxisY.Maximum = 25;  chart1.ChartAreas[0].AxisX.MajorGrid.Interval = 0.5;  chart1.Series[0].Points.DataBindXY(x, y);  chart1.Series[1].Color = Color.FromArgb(255, 0, 0);  chart1.Series[1].ChartType = SeriesChartType.Point;  chart1.Series[1].BorderWidth = 2;  for (int i = 0; i < x.Length; i++)  {  chart1.Series[1]["PixelPointWidth"] = "15";  chart1.Series[1].Points.AddXY(x[i], y[i]);  }  }  private void Graph(double[] x, double[] y, double[] pX, double[] pF)  {  chart1.Series[1].Points.Clear();  chart1.ChartAreas[0].AxisX.Minimum = -5;  chart1.ChartAreas[0].AxisX.Maximum = 10;  chart1.ChartAreas[0].AxisY.Minimum = -10;  chart1.ChartAreas[0].AxisY.Maximum = 20;  chart1.Series[0].Name = "Spline";  chart1.Series[0].Color = Color.FromArgb(0, 0, 0);  chart1.ChartAreas[0].AxisX.MajorGrid.Interval = 0.5;  chart1.Series[0].Points.DataBindXY(x, y);  chart1.Series[1].Color = Color.FromArgb(255, 0, 0);  chart1.Series[1].ChartType = SeriesChartType.Point;  chart1.Series[1].BorderWidth = 2;  for (int i = 0; i< 5; i++)  {  chart1.Series[1]["PixelPointWidth"] = "25";  chart1.Series[1].Points.AddXY(pX[i], pF[i]);  }  }  //функция считает производные по аргументу x и значению f  private double[] diff(double[] x, double[] f)  {  int n = x.Length;  double[] dif = new double[n];  // Рассчитываем производную  for (int i = 0; i < n - 1; i++)  {  dif[i] = (f[i + 1] - f[i]) / (x[i + 1] - x[i]);  }  // Для последней точки можно использовать аналогичное разностное приближение  dif[n - 1] = dif[n - 2]; // Например, повторяем значение производной в предыдущей точке  return dif;  }  //построить первую производную  private void button2\_Click(object sender, EventArgs e)  {  if (!checkTable())  return;  // Заполняем массивы из текстбоксов  double[] x = array(X, n);  double[] f = array(F, n);  // Вычисляем производную с использованием метода конечных разностей  double[] dif = diff(x, f);  // Строим график  graph(x, dif);    // Выводим значения производных для проверки  textBox20.Text = "f' = ";  for (int i = 0; i < n; i++)  {  textBox20.Text += dif[i].ToString() + " ";  }  }  //построить вторую производную  private void button3\_Click(object sender, EventArgs e)  {  if (!checkTable())  return;  //заполняем массивы из текстбоксов  double[] x = array(X, n);  double[] f = array(F, n);  double[] dif, dif2;  dif = diff(x, f); //первая производная  dif2 = diff(x, dif); //считаем вторую производную по первой  //строим график  graph(x, dif2);    //выводим значения вторых производных для проверки  textBox20.Text = "f'' = ";  for (int i = 0; i < n; i++)  {  textBox20.Text += dif2[i].ToString() + " ";  }  }  //метод прогонки  private double[] TridiagonalMatrixAlgorithm(double[,] matrix, double[] y, int n)  {  double[] x = new double[n];  double[] alpha = new double[n];  double[] beta = new double[n];  // Прямой ход: вычисление прогоночных коэффициентов  alpha[0] = matrix[0, 0];  beta[0] = y[0] / alpha[0];  for (int i = 1; i < n; i++)  {  double denom = matrix[i, i] - matrix[i, i - 1] \* matrix[i - 1, i] / alpha[i - 1];  alpha[i] = denom;  beta[i] = (y[i] - matrix[i, i - 1] \* beta[i - 1]) / denom;  }  // Обратный ход: вычисление решения  x[n - 1] = beta[n - 1];  for (int i = n - 2; i >= 0; i--)  {  x[i] = beta[i] - matrix[i, i + 1] \* x[i + 1] / alpha[i];  }  return x;  }    private void button4\_Click(object sender, EventArgs e)  {  if (!checkTable())  return;  //заполняем массивы из текстбоксов  double[] x = array(X, n);  double[] f = array(F, n);  double[] h = new double[n];  //коэффициенты  double[] a = new double[n - 1], b = new double[n - 1], c = new double[n - 1], d = new double[n - 1];  double[,] matrixC = new double[n - 2, n - 2];  double[] beta = new double[n - 2]; //массив правых частей для СЛАУ  double[,] tempC = new double[n - 2, n];  double min, max;  double[] xGauss = new double[n - 2]; //решение СЛАУ  for (int i = 0; i < n - 1; i++)  a[i] = f[i];  h[0] = x[1] - x[0];  for (int i = 1; i < n - 1; i++)  h[i] = x[i + 1] - x[i];  c[0] = 0;  //заполняем вспомогательную матрицу нулями  for (int i = 0; i < n - 2; i++)  for (int ii = 0; ii < n; ii++)  tempC[i, ii] = 0;  //заполняем вспомогательную матрицу коэффициентов tempС  int k = 1,t=1;  for (int i = 0; i < n - 2; i++)  {  tempC[i, k] = 2 \* (h[k-1] + h[k]);  tempC[i, k - 1] = h[k - 1];  tempC[i, k + 1] = h[k];  k++;  }    //делаем сдвиг и формируем матрицу коэффициентов для решения СЛАУ  for (int i = 0; i < n - 2; i++)  for (int j = 0; j < n - 2; j++)  matrixC[i, j] = tempC[i, j + 1];  //заполняем массив значений правой части СЛАУ  k = 2;  for (int i = 0; i < n - 2; i++)  {  beta[i] = 3 \* ((f[k] - f[k - 1]) / h[t] - (f[k - 1] - f[k - 2]) / h[t - 1]);  k++;  t++;  }  xGauss = TridiagonalMatrixAlgorithm(matrixC, beta, n - 2);  for (int i = 1; i < n - 1; i++)  c[i] = Math.Round(xGauss[i - 1],3);  //считаем b, d при c[n-2]=0  d[n-2] = Math.Round((0 - c[n-2]) / (3 \* h[n-2]),3);  b[n-2] = Math.Round((f[n-1] - f[n-2]) / h[n-2] - (0 + 2 \* c[n-2]) \* h[n-2] / 3,3);  //считаем остальниые b, d  k = 1;  for(int i=0;i<n-2;i++)  {  d[i] = Math.Round((c[i + 1] - c[i]) / (3 \* h[i]),3);  b[i] = Math.Round((f[k] - f[k - 1]) / h[i] - (c[i + 1] + 2 \* c[i]) \* h[i] / 3,3);  k++;  }  //выводим коэффициенты для проверки  textBox20.Text += Environment.NewLine + "a: ";  for (int i = 0; i < n - 1; i++)  textBox20.Text += a[i] + " ";  textBox20.Text += Environment.NewLine+"c: ";  for (int i = 0; i < n - 1; i++)  textBox20.Text += c[i] + " ";  textBox20.Text += Environment.NewLine+"b: ";  for (int i = 0; i < n - 1; i++)  textBox20.Text += b[i] + " ";  textBox20.Text += Environment.NewLine+"d: ";  for (int i = 0; i < n - 1; i++)  textBox20.Text += d[i] + " ";  int[] count = new int[n - 1];  int sum = 0;  for (int i = 0; i < n - 1; i++)  {  //границы интервалов  min = x[i];  max = x[i + 1];  count[i] = (int)Math.Ceiling((max - min) / step) + 1; //количество точек участка графика  if (i == 0)  sum += count[i];  else  sum += count[i] - 1;  }  int temp = 0;  textBox20.Text += Environment.NewLine;  double[] abs = new double[sum];  double[] ord = new double[sum];  int tmp=0;  //считаем функцию сплайна на каждом интервале, всего интервалов n-1  for (int j = 0; j < n - 1; j++)  {  //границы интервалов  min = x[j];  if (j != 0)  tmp = 1;  for (int i = tmp; i < count[j]; i++)  {  abs[temp] = min + step \* i;  ord[temp] = a[j] + b[j] \* (abs[temp] - x[j]) + c[j] \* Math.Pow(abs[temp] - x[j], 2) + d[j] \* Math.Pow(abs[temp] - x[j], 3);  temp++;  }  }  Graph(abs, ord,x,f);    }  private double func(double x)  {  return 2 \* x \* x;  }  private void button5\_Click(object sender, EventArgs e)  {  // Задаем точки, в которых будем вычислять производные  double[] xValues = { Math.Pow(10, -2), Math.Pow(10, -1), 1, 10, Math.Pow(10, 2) };  // Задаем функцию f(x) = 2 \* x^2  double[] fValues = new double[xValues.Length];  for (int i = 0; i < xValues.Length; i++)  {  fValues[i] = 2 \* Math.Pow(xValues[i], 2);  }  // Вызываем функцию для вычисления приближенных производных с различными шагами  CalculateDerivatives();  }  void CalculateDerivatives()  {  double x = 0.01;  double[] xValues = { Math.Pow(10, -2), Math.Pow(10, -1), 1, 10, Math.Pow(10, 2) };  // Выполняем вычисления для различных шагов  for (int step = 1; step <= 8; step++)  {  double h = Math.Pow(10, -step);  richTextBox1.Text += $"Шаг равен: {h}" + Environment.NewLine;  for(int i = 0; i < 5; i++)  {  double fx = 0;  fx = (func(xValues[i]) - func(xValues[i] - h)) / h;  richTextBox1.Text += $"Точка {xValues[i]}, Разница {Math.Abs(fx - 4 \* xValues[i])}" + Environment.NewLine;  }  }  }  }  } |