МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

**

**ОТЧЁТ**

**по лабораторной работе №3**

**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. Для функции заданной таблицей значений согласно варианту задания из таблицы 3.2 в любом доступном математическом программном пакете выполнить:

- расчет коэффициентов сглаживающих многочленов степеней 1, 2, 3 по методу наименьших квадратов;

- расчет коэффициентов интерполяционного многочлена с помощью решения системы уравнений (3.1);

- построение графиков полученных многочленов.

2. Разработать программную реализацию построения графиков интерполяционного многочлена, полученного с помощью формулы Лагранжа, по произвольной таблице значений функции; интерполяционного многочлена, полученного с помощью формулы Ньютона, по произвольной таблице значений функции; сглаживающих многочленов 1, 2, 3 степени, полученных по методу наименьших квадратов, по произвольной таблице значений функции; произвольного многочлена четвертой степени, заданного набором коэффициентов (для выполнения следующего пункта задания). При разработке алгоритма стараться по возможности минимизировать вычислительные затраты. Для решения системы (3.2) использовать реализацию метода Гаусса с выбором главного элемента из соответствующей лабораторной работы. Графики отображать как кусочно-линейную аппроксимацию с малым (~10-2) шагом. Стили отображения линий (штриховка, цвет, толщина и т.п.) должны позволять видеть совпадающие графики. Желательно отдельно отобразить точки исходной таблицы.

3. С помощью разработанной программы показать совпадение графиков многочленов, рассчитанных вручную в пункте 1 с реализованными программно.

**Исходные данные**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | -1 | 1 | 2 | 3 | 4 |
|  | -5 | -3 | 18 | 6 | -2 |

**Ход работы**

Для успешного выполнения работы мы рассчитали коэффициенты сглаживающих многочленов по методу наименьших квадратов:

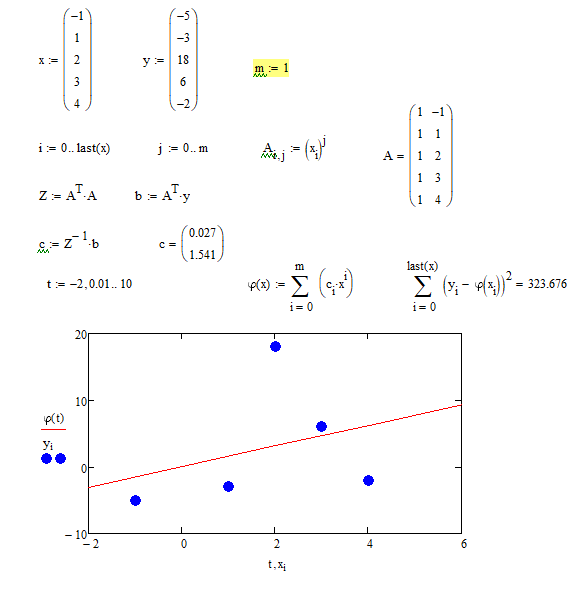


Рис. 1 - Линейный МНК

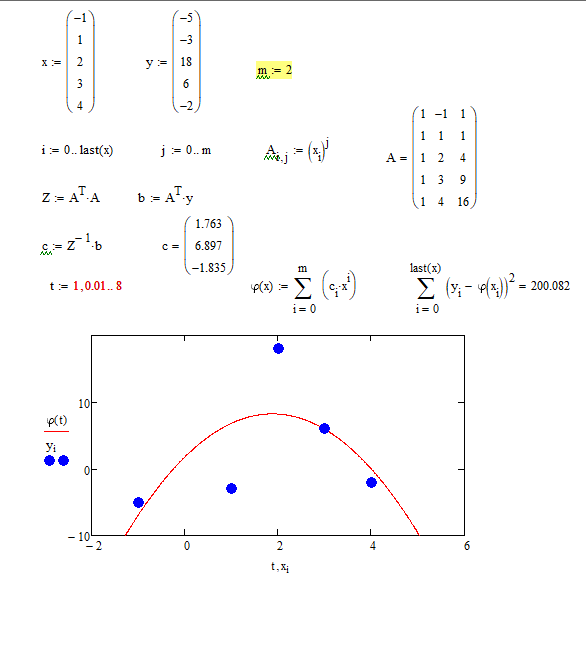


Рис. 2 - Квадратичный МНК

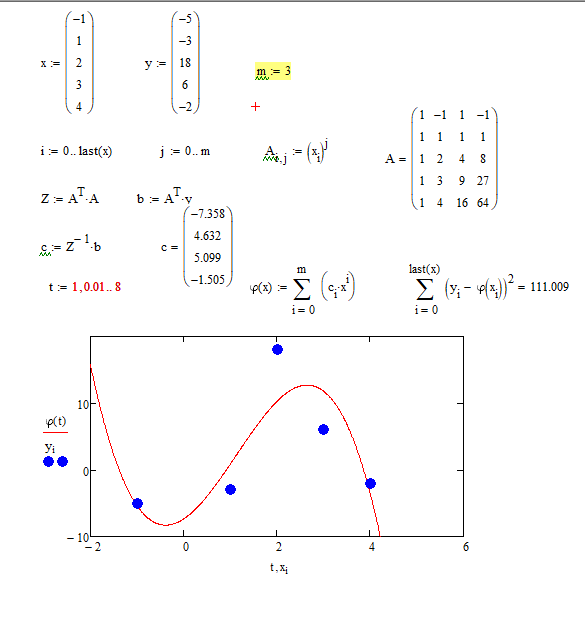
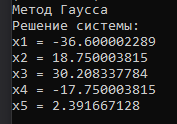
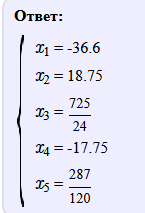


Рис. 3 - Кубический МНК

Затем рассчитали коэффициенты интерполяционного многочлена:

На основе теории мы реализовали программу:

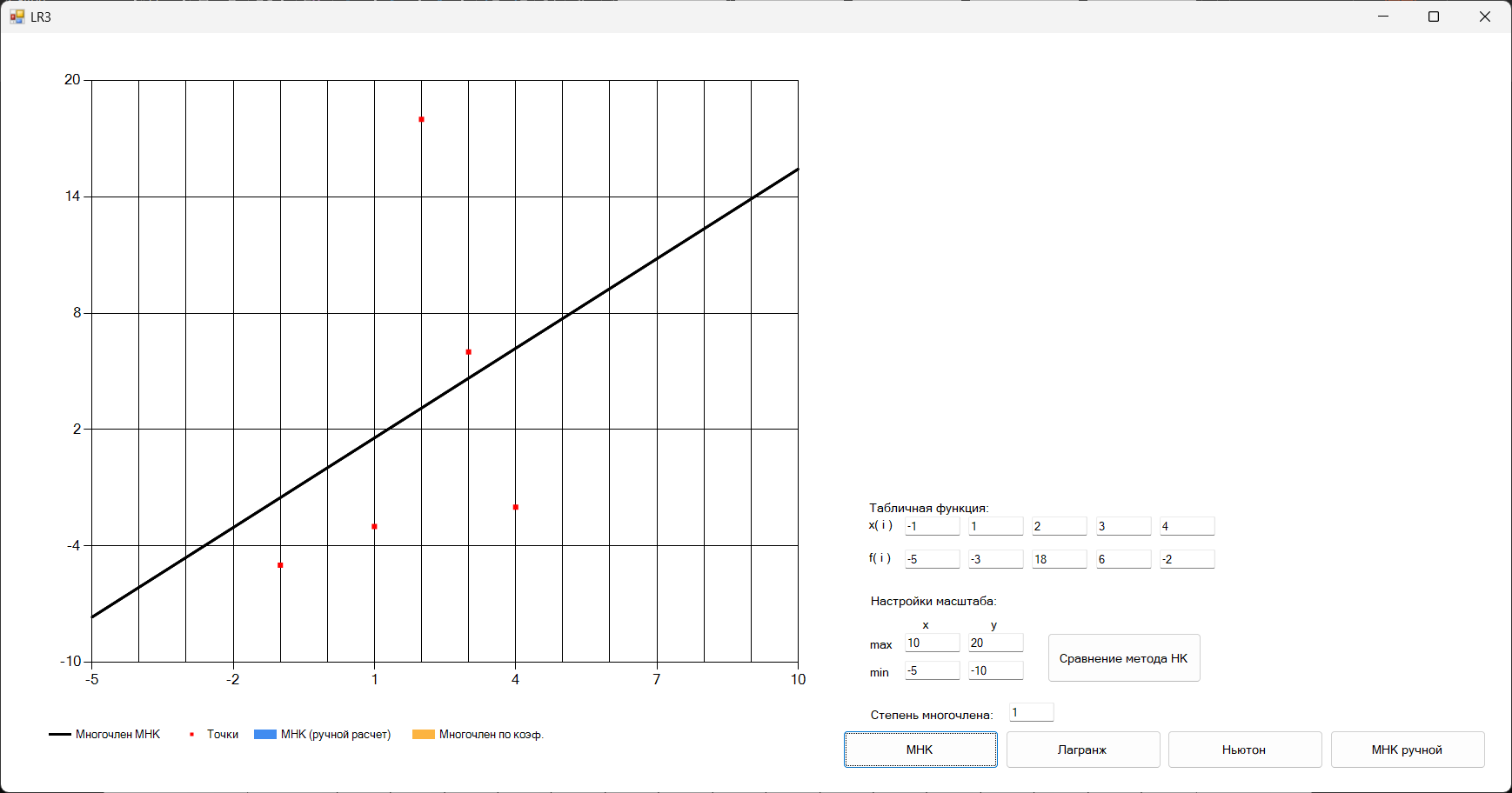


Рис. 4 - Многочлен по МНК 1 степени (программный)

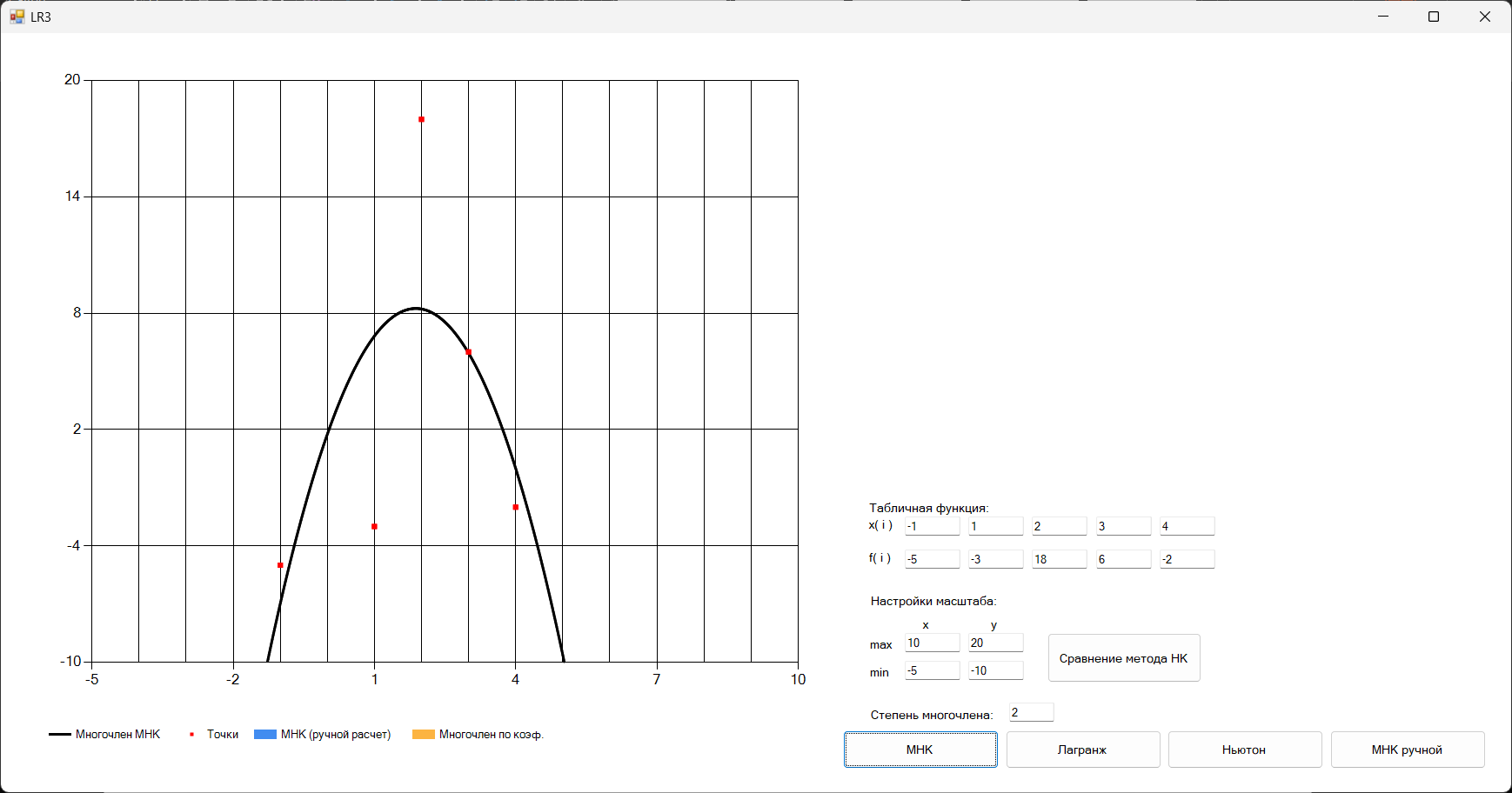


Рис. 5 - Многочлен по МНК 2 степени (программный)

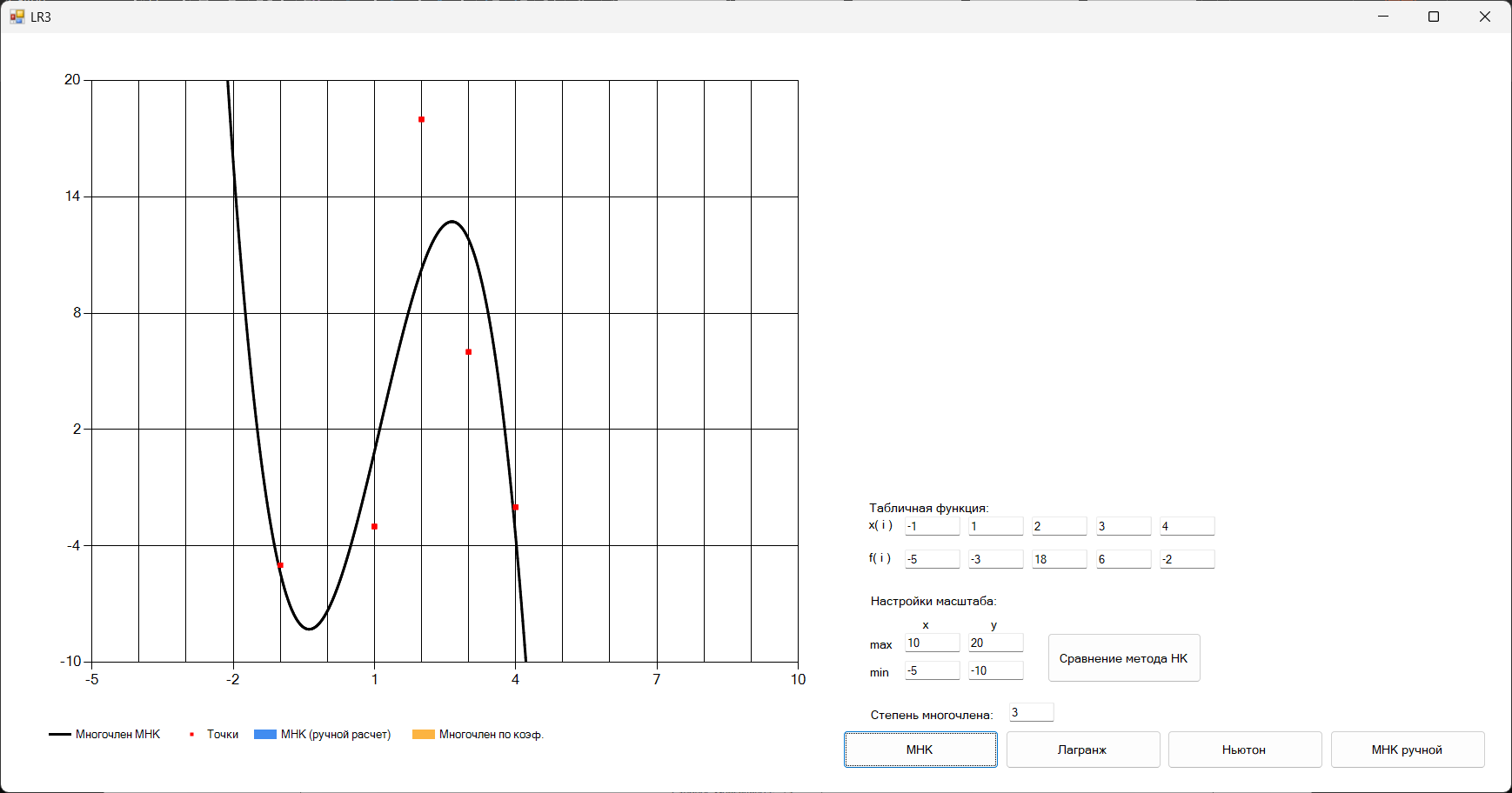


Рис. 6 - Многочлен по МНК 3 степени (программный)

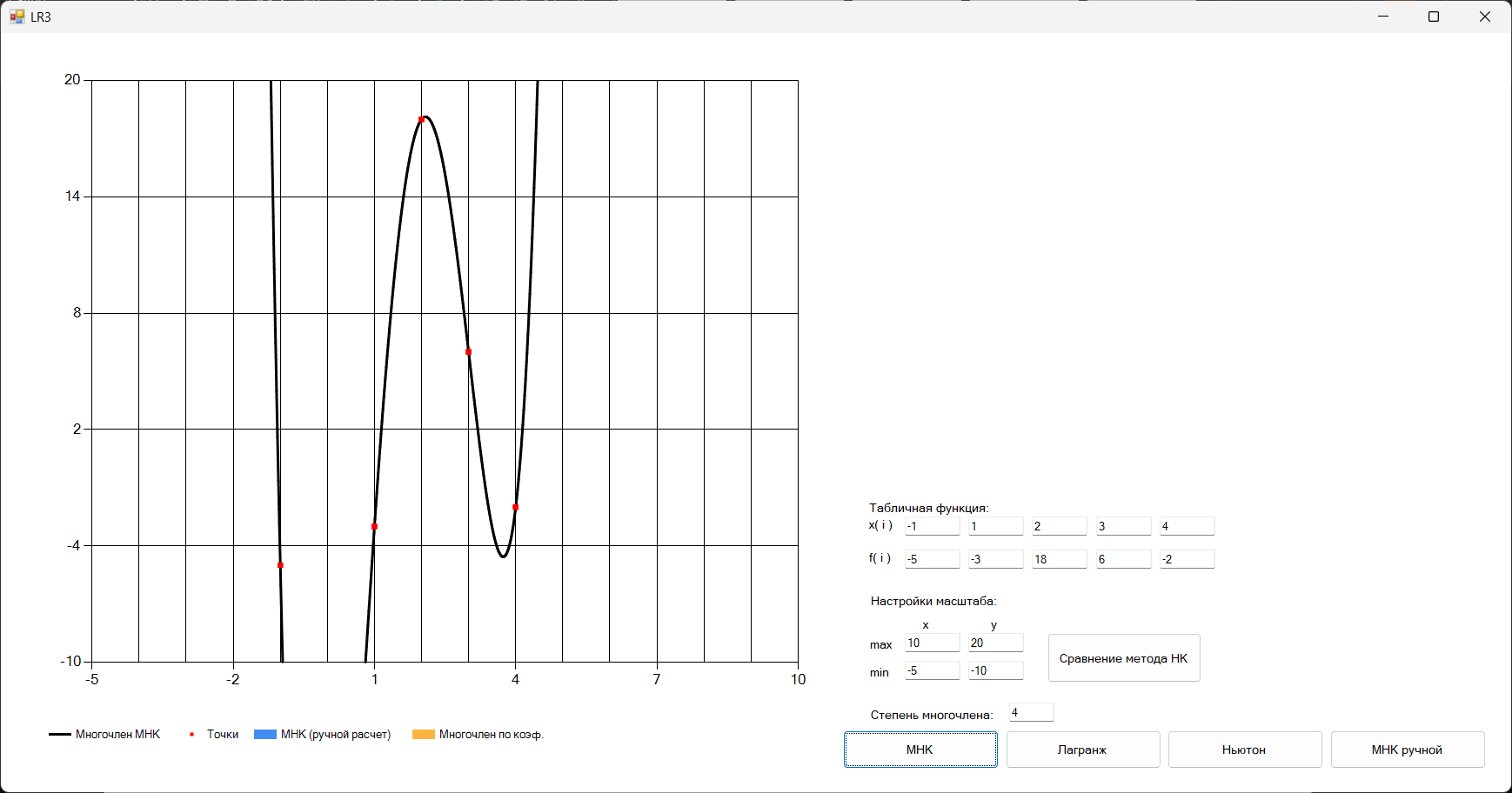


Рис. 7 - Многочлен по МНК 4 степени (программный)

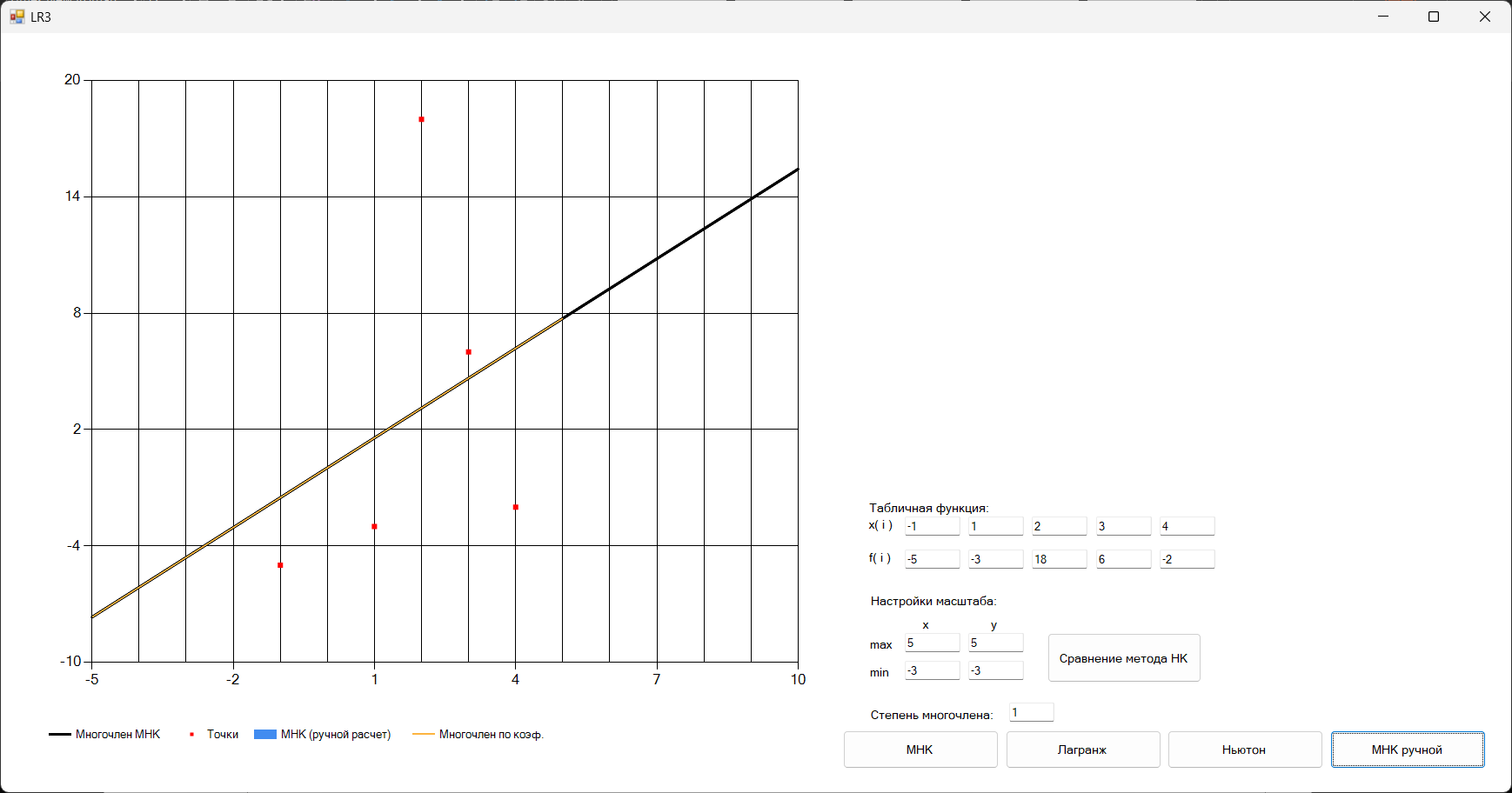


Рис. 8 - Сравнение графика МНК (черный) и графика многочлена по коэф. (желтый) 1 степени

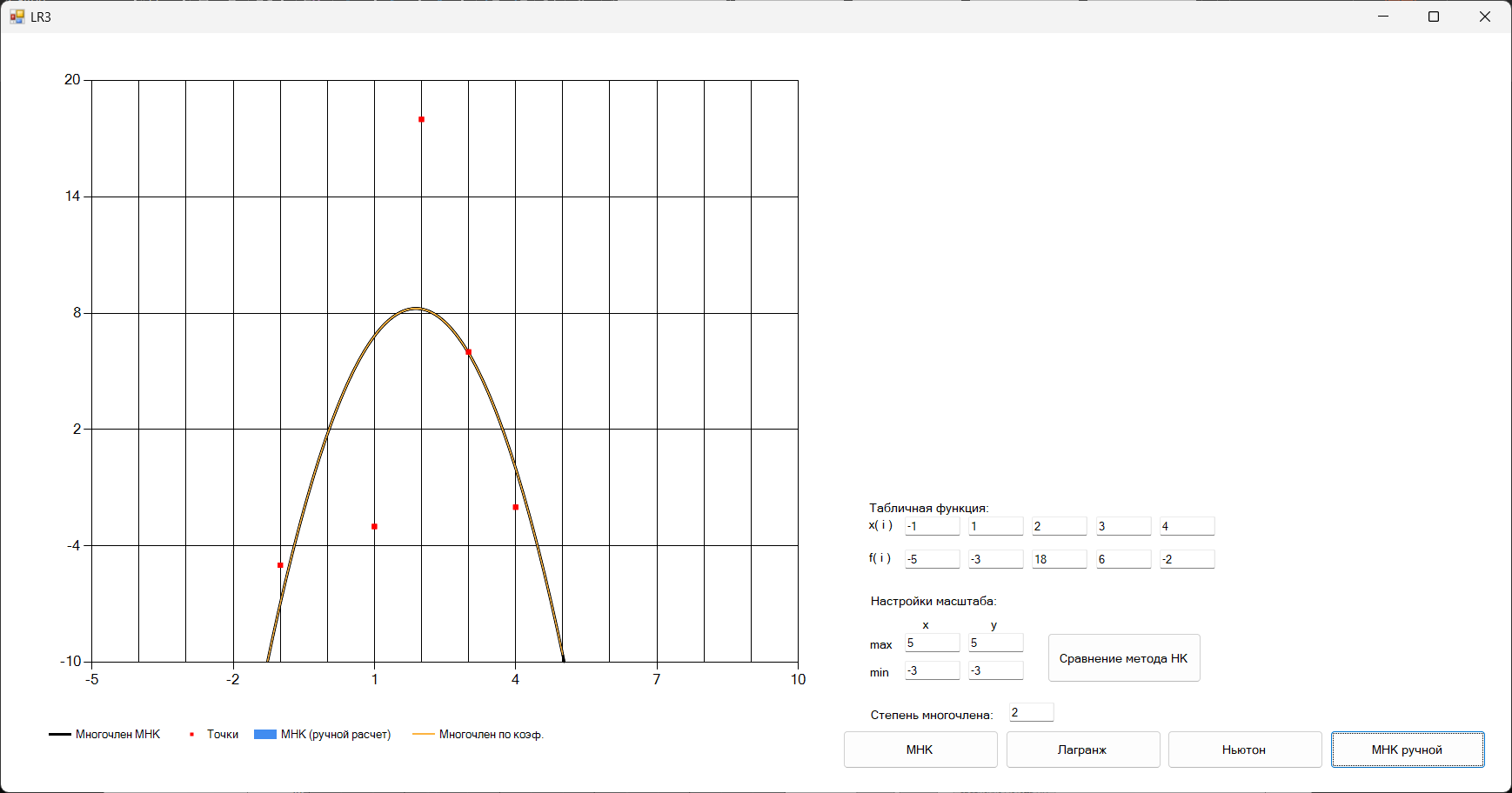


Рис. 9 - Сравнение графика МНК (черный) и графика многочлена по коэф. (желтый) 2 степени

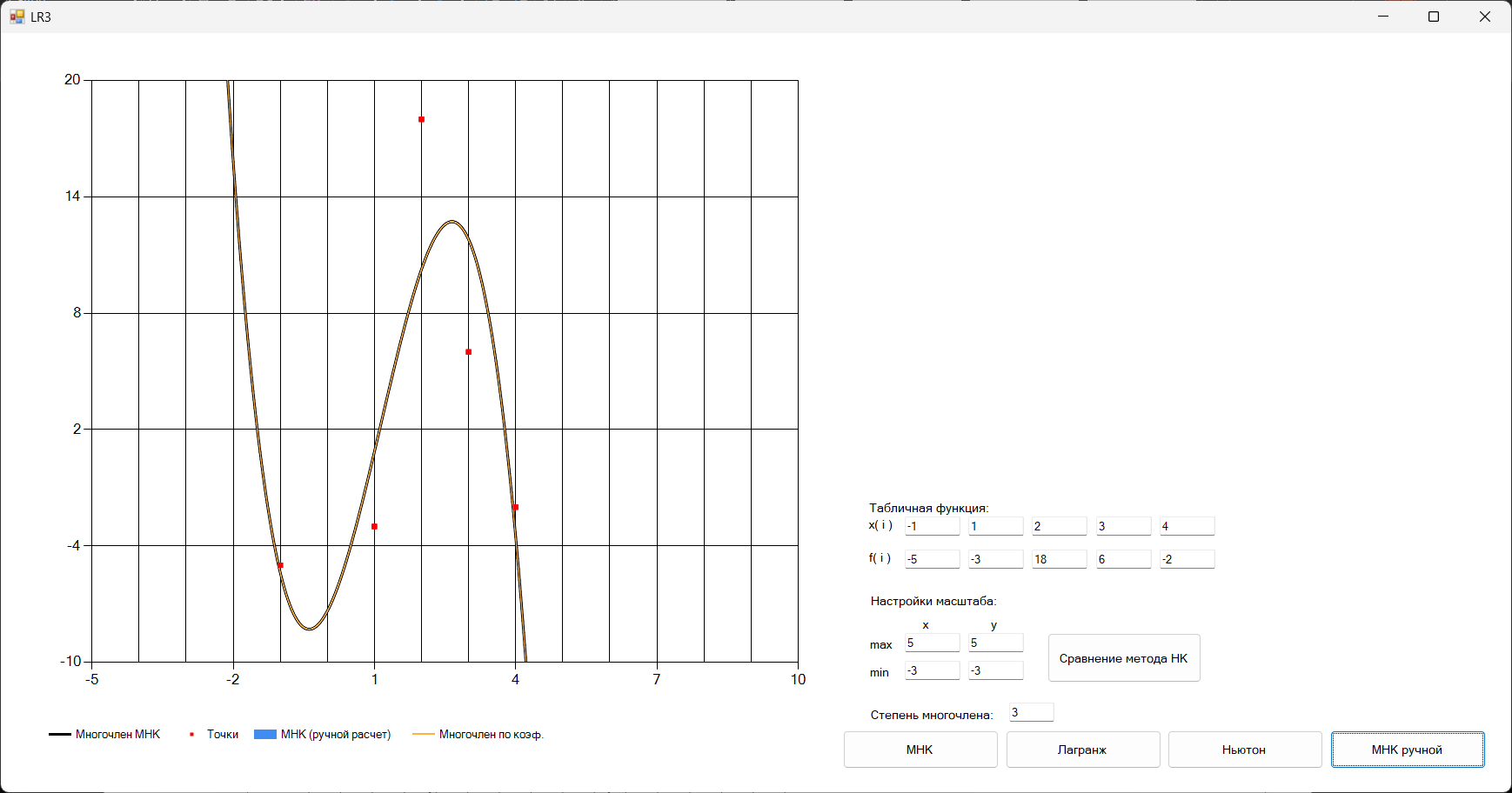


Рис. 10 - Сравнение графика МНК (черный) и графика многочлена по коэф. (желтый) 3 степени

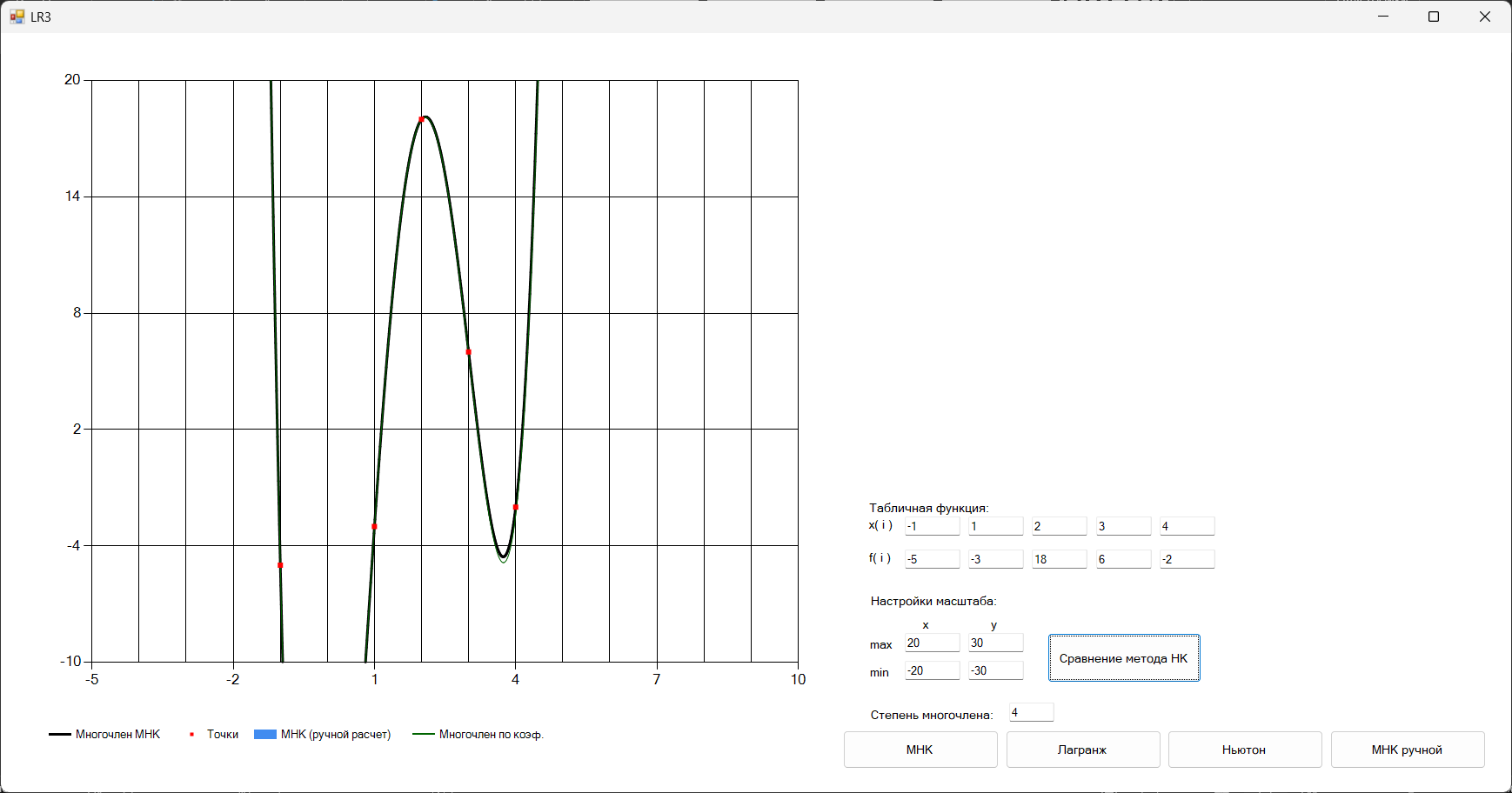


Рис. 11.1 - Сравнение графика МНК (черный) с графиком многочлена по коэффициентам (зеленый)

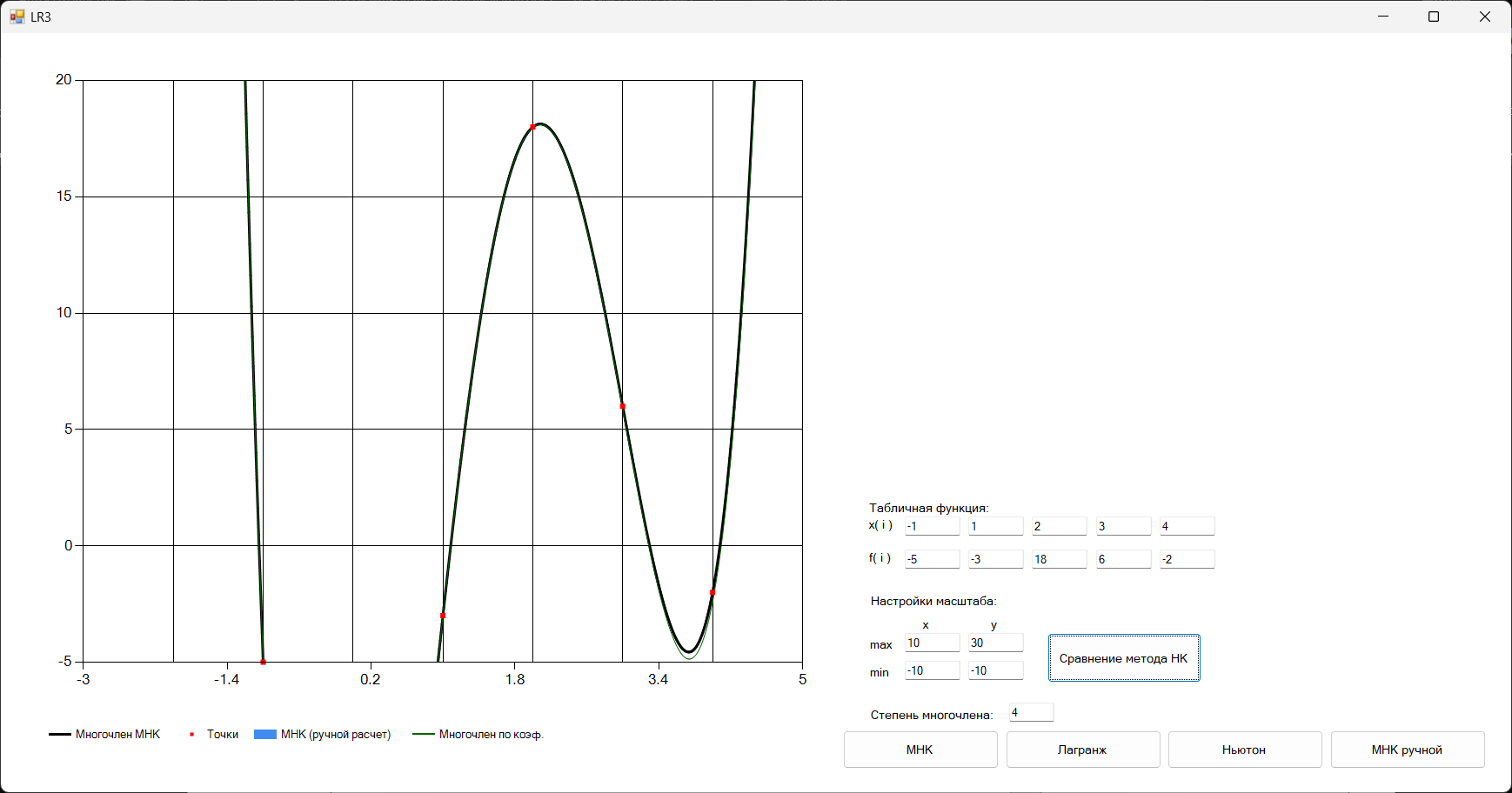


Рис. 11.2 - Сравнение графика МНК (черный) с графиком многочлена по коэффициентам (зеленый)

Графики, рассчитанные в программе, оказались точнее, чем график многочлена, построенного по коэффициентам. Это связано с тем, что при расчете этих коэффициентов мы не учли точность вычислений, в связи с чем отсутствие, пусть и незначительных на первый взгляд, 3-4 дополнительных цифр после запятой сказалось на качестве построенного нами графика. Виднее всего данная неточность проявляется в местах изгиба графика.

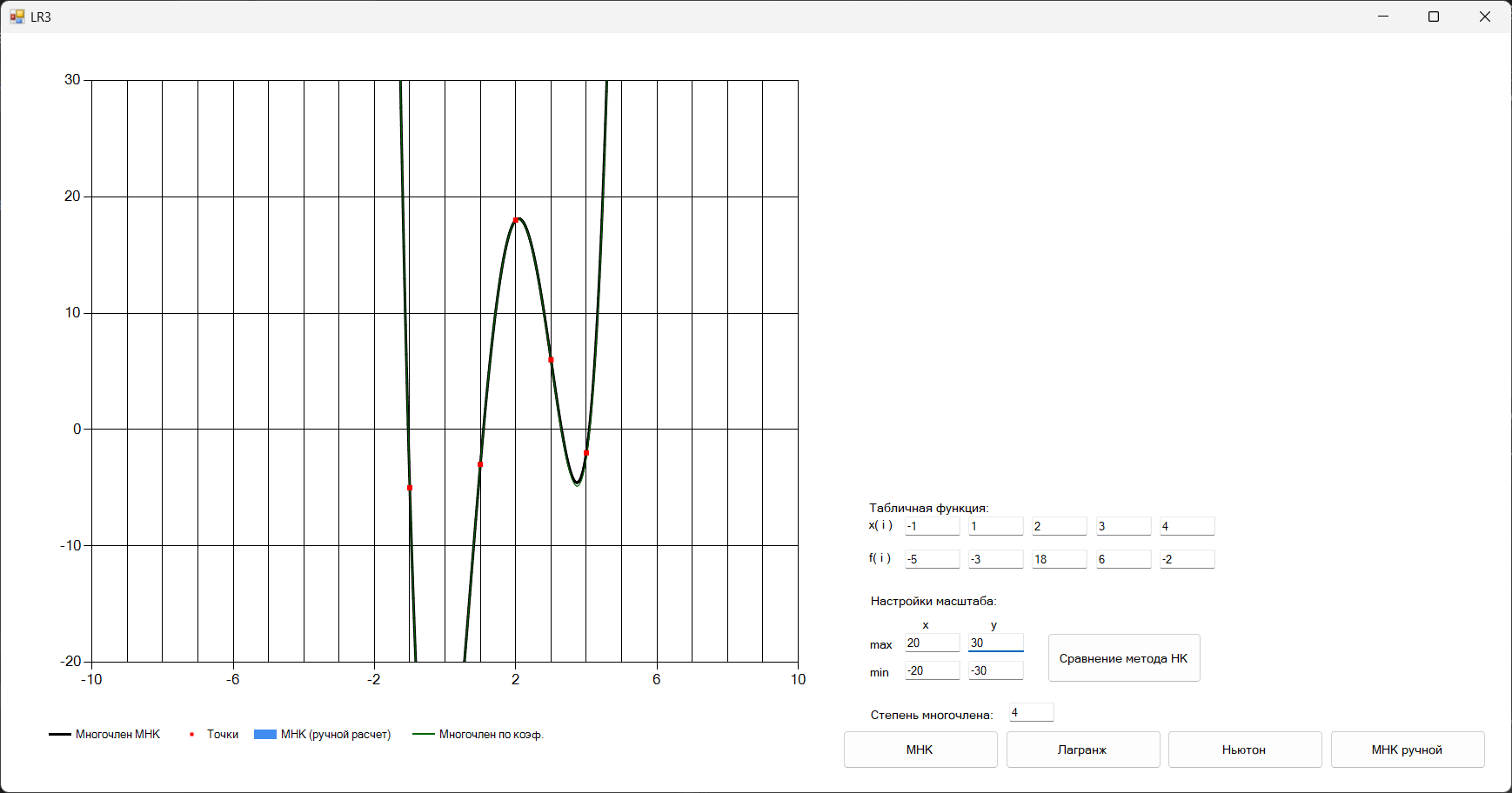


Рис. 12 - Сравнение графика Лагранжа с графиком многочлена по коэффициентам

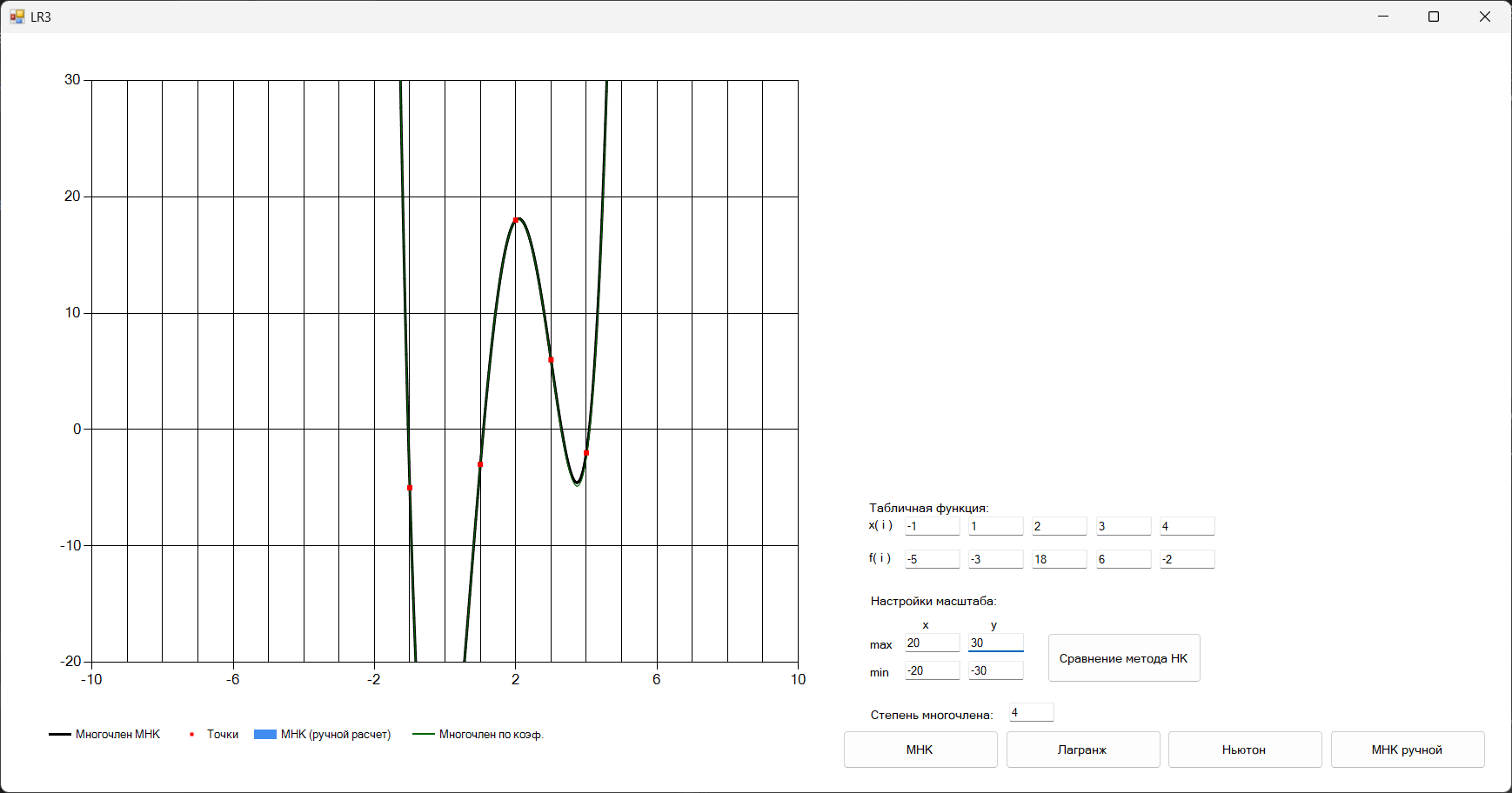


Рис. 13 - Сравнение графика Ньютона с графиком многочлена по коэффициентам

**Объяснение полученных результатов**

В данной лабораторной работе были реализованы такие методы, как: построение сглаживающих многочленов 1, 2 и 3 степеней, и построение интерполяционного многочлена по формулам Лагранжа и Ньютона.

Построение сглаживающих многочленов намного проще, чем интерполяционного многочлена.

Так получается из-за того, что для сглаживающих многочленов всегда одна и та же СЛАУ, вместо конкретных x и y из таблицы, используется их сумма, а значит в программе не нужно будет строить для каждого х свою систему, стоит лишь решить систему один раз, найти коэффициенты многочлена и сразу получить рабочую формулу, куда остаётся только подставлять значения х. С интерполяционным многочленом дела обстоят иначе: для каждого х приходится заново считать все значения, потому что после записи формулы нужно упростить многочлен, для получения его коэффициентов при х.

**Код разработнной программы**

|  |
| --- |
| main |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.Windows.Forms.DataVisualization.Charting;  namespace vchmat3  {      public partial class Form1 : Form      {          public Form1()          {              InitializeComponent();          }          int n;          TextBox[] X = new TextBox[7];  // Массив ячеек x(i)          TextBox[] F = new TextBox[7];  // Массив ячеек f(i)          double step = 1;          // Создание массива с данными из текстбокса          private double[] array(TextBox[] box, int n)          {              double[] x = new double[n];              double[] f = new double[n];              //заполняем массив x, f из таблицы              for (int i = 0; i < n; i++)              {                  x[i] = new double();                  x[i] = double.Parse(box[i].Text);              }              return x;          }          // Проверка, что не все х=0          private bool check(TextBox[] box, int n)          {              int fl = 0;              for (int i = 0; i < n; i++)                  if (double.Parse(box[i].Text) == 0)                      fl++;              if (fl == n)              {                  MessageBox.Show("Невозможно построить график. Введите другие значения х.");                  return false;              }              else                  return true;          }          //метод Гаусса без выбора главного элемента          private double[] gauss(double[,] matrix, double[] y, int n)          {              double[] x;              x = new double[n];              // переставим строки так, чтобы диагоналные элементы были не 0              for (int ind = 0; ind < n; ind++)              {                  int numb = ind;                  for (int i = 1; i < n; i++)                      if (matrix[i, ind] != 0)                          numb = i;                  //перестановка строк,ставим на позицию ind строку, в которой ind элемент max                  if (numb != ind)                  {                      //идем по строке                      for (int i = 0; i < n; i++)                      {                          double tempp = matrix[ind, i];                          matrix[ind, i] = matrix[numb, i];                          matrix[numb, i] = tempp;                      }                      double temp = y[ind];                      y[ind] = y[numb];                      y[numb] = temp;                  }              }              //идем по переменным              for (int ind = 0; ind < n; ind++)              {                  //приведем расширенную матрицу к ступенчатому виду                  //идем по строкам,начиная со следующей после выбранной                  for (int i = ind + 1; i < n; i++)                  {                      double mult = -matrix[i, ind] / matrix[ind, ind];                      if (matrix[i, ind] != 0)                      {                          for (int j = ind; j < n; j++)                              matrix[i, j] += matrix[ind, j] \* mult;                      }                      else                          continue;                      y[i] += y[ind] \* mult;                  }              }              //обратная подстановка              for (int i = n - 1; i >= 0; i--)              {                  x[i] = y[i] / matrix[i, i];                  for (int j = 0; j < i; j++)                      y[j] = y[j] - matrix[j, i] \* x[i];              }              return x;          }          // Сглаживающй мнгочлен 1 степени          private void kv1()          {              chart1.Series[3].Points.Clear();              chart1.Series[3].ChartType = SeriesChartType.Line;              double MCH, x;              for (x = -5; x <= 5; x += 0.001)              {                  MCH = 0.027 + 1.541 \* x;                  chart1.Series[3].Points.AddXY(x, MCH);              }          }          // Сглаживающй мнгочлен 2 степени          private void kv2()          {              chart1.Series[3].Points.Clear();              chart1.Series[3].ChartType = SeriesChartType.Line;              double MCH, x;              for (x = -5; x <= 5; x += 0.001)              {                  MCH = -Math.Pow(x, 2) \* 1.835 + 6.897 \* x + 1.763;                  chart1.Series[3].Points.AddXY(x, MCH);              }          }          // Сглаживающй мнгочлен 3 степени          private void kv3()          {              chart1.Series[3].Points.Clear();              chart1.Series[3].ChartType = SeriesChartType.Line;              double MCH, x;              for (x = -5; x <= 5; x += 0.001)              {                  MCH = -Math.Pow(x, 3) \* 1.505 - Math.Pow(x, 2) \* 5.099 + 4.632 \* x - 7.358;                  chart1.Series[3].Points.AddXY(x, MCH);              }          }          // Интерполяционный мнгочлен с коэффициентами          private double FUNC(double x)          {              double interpolated;              interpolated = -36.6 + 18.75 \* x + 30.21 \* Math.Pow(x, 2) - 17.75 \* Math.Pow(x, 3) + 2.39 \* Math.Pow(x, 4);              return interpolated;          }          // Метод наименших квадратов          private void button2\_Click(object sender, EventArgs e)          {              if (check(X, n) == false)              {                  cleaning(n);                  return;              }              else                   if (int.Parse(textBox16.Text) == 0)              {                  MessageBox.Show("Невозможно построить график. Степень многочлена не должна равняться 0.");                  cleaning(n);                  return;              }              //заполняем массивы из полей для ввода              double[] x = array(X, n);              double[] f = array(F, n);              int.TryParse(textBox16.Text, out int k); // Степень многочлена              double[] a = new double[k + 1];              double[,] c = new double[k + 1, k + 1];              double[] d = new double[k + 1];              int m = 2 \* k;              double sum;              double[] C = new double[m + 1];              if (k == 0)              {                  MessageBox.Show("Невозможно построить график. Степень многочлена не должна равняться 0.");                  return;              }              // Считаем С              for (int i = 0; i <= m; i++)              {                  C[i] = new double();                  sum = 0;                  for (int j = 0; j < n; j++)                      sum += Math.Pow(x[j], i);                  C[i] = sum;              }              // Записываем двумерный массив С для решения СЛАУ              int temp;              for (int i = 0; i < k + 1; i++)              {                  temp = i;                  for (int j = 0; j < k + 1; j++)                  {                      c[i, j] = new double();                      c[i, j] = C[temp];                      temp++;                  }              }              // Считаем D и заполняем массив              for (int i = 0; i < k + 1; i++)              {                  sum = 0;                  d[i] = new double();                  for (int j = 0; j < n; j++)                      sum += f[j] \* Math.Pow(x[j], i);                  d[i] = sum;              }              // Решим СЛАУ методом Гаусса и получим коэф. аппроксимирующего многочлена              a = gauss(c, d, k + 1);              // Строим график              double xMin = x[0] \* 5;              double xMax = x[n - 1] \* 5;              int count = (int)Math.Ceiling((xMax - xMin) / step) + 1;              double[] abs = new double[count];              double[] ord = new double[count];              for (int i = 0; i < count; i++)              {                  abs[i] = xMin + step \* i;                  for (int j = 0; j <= k; j++)                      ord[i] += Math.Pow(abs[i], j) \* a[j];              }              graph(xMin, xMax, step, abs, ord, x, f);          }          // Метод Лагранжа          private void button3\_Click(object sender, EventArgs e)          {              if (check(X, n) == false)              {                  cleaning(n);                  return;              }              //заполняем массивы из полей для ввода              double[] x = array(X, n);              double[] f = array(F, n);              // Строим график              double xMin = x[0] \* 5;              double xMax = x[n - 1] \* 5;              int count = (int)Math.Ceiling((xMax - xMin) / step) + 1;              double[] abs = new double[count];              double[] ord = new double[count];              double L;              double num, den; // Числитель и знаменатель для вычисления многочлена в форме Лагранжа              for (int i = 0; i < count; i++)              {                  abs[i] = xMin + step \* i;                  L = 0;                  for (int j = 0; j < n; j++)                  {                      num = 1;                      den = 1;                      for (int ii = 0; ii < n; ii++)                      {                          if (ii != j)                          {                              num \*= abs[i] - x[ii];                              den \*= x[j] - x[ii];                          }                          else                              continue;                      }                      L += f[j] \* num / den;                  }                  ord[i] = L;              }              Graph(xMin, xMax, step, abs, ord);          }          // Метод Ньютона          private void button4\_Click(object sender, EventArgs e)          {              if (check(X, n) == false)              {                  cleaning(n);                  return;              }              // Заполняем массивы из текстбоксов              double[] x = array(X, n);              double[] f = array(F, n);              // Строим график              double xMin = x[0] \* 5;              double xMax = x[n - 1] \* 5;              int count = (int)Math.Ceiling((xMax - xMin) / step) + 1;              double[] abs = new double[count];              double[] ord = new double[count];              double[,] ff = new double[n, n];              for (int i = 0; i < n; i++)                  ff[0, i] = f[i];              int k = n - 1;              // Разделенные разности порядка i              for (int i = 1; i < n; i++)              {                  int a = 0;                  for (int j = 0; j < k; j++)                  {                      ff[i, j] = (ff[i - 1, j + 1] - ff[i - 1, j]) / (x[i + a] - x[j]);                      a++;                  }                  k--;              }              // Строим график              double P;              double mult;              for (int i = 0; i < count; i++)              {                  P = ff[0, 0];                  abs[i] = xMin + step \* i;                  for (int j = 1; j < n; j++)                  {                      k = j - 1;                      mult = 1;                      while (k >= 0)                      {                          mult \*= abs[i] - x[k];                          k--;                      }                      P += ff[j, 0] \* mult;                  }                  ord[i] = P;              }              Graph(xMin, xMax, step, abs, ord);          } |