Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ “Информатика и системы управления”

КАФЕДРА "Программное обеспечение ЭВМ и информационные технологии”

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

НА ТЕМУ:

“Алгоритм и программная реализация одномерной линейной интерполяции сеточной функции кубическими сплайнами”

Студент ИУ7-53Б(В) _____ М. А. Сергеев
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ д.т.н. В. М. Градов
(Подпись, дата) (И.О.Фамилия)

Москва
2022 г.

Оглавление

1. Аналитический раздел

Целью научно-исследовательской работы является изучение и программная реализация алгоритма одномерной линейной интерполяции сеточной функции кубическими сплайнами, а также графическое представление полученных результатов моделирования.

Под одномерной линейной интерполяцией понимается интерполяция алгебраическим двучленом $P_1(x) = ax + b$ функции одной переменной $f(x)$, заданной в двух точках x_0 и x_1 отрезка $[a, b]$.

Слово «сплайн» можно перевести как «гибкая линейка». Такую линейку можно использовать для проведения кривых через заданную совокупность точек, изгибая и придерживая её так, чтобы ребро проходило через все точки на плоскости. Благодаря такому свойству сплайны достаточно давно используются в задачах геометрического моделирования, например, задания обводов корпусов водных и воздушных судов в машиностроительном черчении. Распространение электронно-вычислительной техники нашло применение им и в задачах математического моделирования. Под кубическим сплайном понимается гладкая функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым полиномом третьей степени.

В реальном мире большое количество физических процессов по самой своей природе являются сплайнами. Таким образом, сплайн - не выдуманная математическая абстракция, а во многих случаях он является решением дифференциальных уравнений, описывающих вполне реальные физические процессы.

Для реализации программы был выбран язык программирования высокого уровня C#/CLI и среда разработки Microsoft Visual Studio Community 2022 с использованием фреймворка .NET. Программа предоставляет пользователю возможность ввода входных данных, выполняет расчет с использованием данных значений и отображает на экране результат в графическом виде.

2. Конструкторский раздел

В ходе заданной в рамках НИРС задачи нужно вычислить значения кубических сплайнов на отрезках сеточной функции. Для каждого такого отрезка между парами соседних точек возможно построение интерполяционного полинома третьей степени:

$$\psi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d(x - x_{i-1})^3,$$
$$x_{i-1} \le x \le x_i, \qquad 0 \le i \le N.$$

Поскольку предполагается, что интерполируемая функция является непрерывной, в узлах значения многочлена и интерполируемой функции совпадают:

$$\psi(x_{i-1}) = y_{i-1} = a_i,$$
$$\psi(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3,$$
$$h_i = \mathrm{x}_i - x_{i-1}, \quad 1 \le i \le N.$$

Число таких уравнений меньше числа неизвестных в два раза. Недостающие уравнения получают, приравнивая во внутренних узлах первые и вторые производные, вычисляемые по коэффициентам на соседних участках:

$$\psi'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$
$$\psi''(x) = 2c_i + 6d_i(x - x_{i-1}), \quad x_{i-1} \le x \le x_i,$$
$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2,$$
$$c_{i+1} = c_i + 3d_i h_i, \quad 1 \le i \le N - 1.$$

Для понимания поведения исходной функции на концах участках интерполирования нам понадобились бы значения функции за его пределами, что противоречит определению интерполирования (поскольку мы находим значения функции внутри данного участка). Однако, мы можем получить недостающие условия, полагая, что вторая производная равна нулю на концах участка интерполирования:

$$\psi''(x_0) = 0, \quad c_1 = 0,$$
$$\psi''(x_N) = 0, \quad c_N + 3d_N h_N = 0.$$

Полученные уравнения позволяют определить все $4N$ неизвестных коэффициентов: a_i, b_i, c_i, d_i $(1 \le i \le N)$. Они могут быть выражены следующим образом:

$$a_i = y_{i-1}, \quad 1 \le i \le N,$$
$$b_i = \frac{y_i - y_{i-1}}{h_i} - h_i \frac{c_{i+1} - 2c_i}{3}, \quad 1 \le i \le N - 1,$$
$$b_N = \frac{y_N - y_{N-1}}{h_N} - h_N \frac{2c_N}{3},$$

$$c_1 = 0,$$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right), \quad 2 \le i \le N-1, \quad (1)$$

$$c_N = -3d_n,$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad 1 \le i \le N-1,$$

$$d_N = -\frac{c_N}{3h_N}.$$

Матрица системы (1) является трёхдиагональной, т.е. все её элементы, кроме находящейся на главной и двух соседних диагоналях, равны нулю. Такие системы удобно решать методом прогонки. Суть метода в следующем.

Система уравнений преобразуется к виду, когда в каждом уравнении содержится только два неизвестных и при обратном ходе одно из них выражается через другое. Тогда можно записать следующее:

$$c_i = \xi_{i+1}c_{i+1} + \eta_{i+1}, \quad (2)$$

где ξ_{i+1}, η_{i+1} – некоторые неизвестные пока прогоночные коэффициенты;

$$c_{i-1} = \xi_i c_i + \eta_i. \quad (3)$$

Подставляя выражение (3) в выражение (1) и преобразуя, получим

$$c_i = -\frac{h_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}c_{i+1} + \frac{f_i - h_{i-1}\eta_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}. \quad (4)$$

Сравнивая выражения (2) и (4), получим

$$\xi_{i+1} = -\frac{h_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}, \quad \eta_{i+1} = \frac{f_i - h_{i-1}\eta_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}. \quad (5)$$

В этих формулах введено обозначение

$$f_i = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right).$$

Из условия $c_1 = 0$ следует $\xi_2 = 0, \mu_2 = 0$.

Теперь алгоритм решения (1) выглядит следующим образом: по формулам (5) при известных ξ_2, η_2, равных нулю, вычисляют прогоночные

коэффициенты ξ_{i+1}, η_{i+1} $(2 \le i \le N)$ [прямой ход]; затем по формулам ❷ при условии $c_{N+1} = 0$ определяют все c_i [обратный ход].

3. Технологический раздел

Для представления координат точек в численном виде в программе используется структура *Point*, содержащая поля *x* и y (оба поля имеют тип double, что позволяет работать с дробными величинами).

```
struct Point
{
    public double x;
    public double y;
}
```

Для хранения координат введённых точек в программе используется список типа *Point*.

```
List<Point> listOfPoints = new List<Point>();
```

Наполнение данного списка происходит следующим образом. В самом начале обработки создаётся переменная p упомянутого выше типа Point. Если в текстовое поле точки 1 введены значения координат X и Y, то запускается обработка введённых значений. Данные значения приводятся к формату double и они присваиваются полям переменной *p*, которая затем добавляется в список хранимых координат функции. В случае, если привести введённое значение к формату double невозможно (например, в любом из текстовых полей введена буква латинского алфавита или две запятых, являющихся отделителем целой части от дробной), будет выведено сообщение об ошибке, информирующее о неверном формате данных введённой точки, а также о факте того, что данная точка при построении графика (в силу невозможности получения координат) использоваться не будет.

```
Point p;
if (textBox1.Text != "" && textBox2.Text != "")
{
try
      {
      p.x = Convert.ToDouble(textBox1.Text);
      p.y = Convert.ToDouble(textBox2.Text);
      listOfPoints.Add(p);
      }
catch
```

```
      {
      MessageBox.Show("Неверный формат записи первой точки координат. " +
      "Координаты должны быть записаны в виде целого числа или десятичной дроби,
      дробная часть которой отделена запятой. Координаты точки 1 [" + textBox1.Text
      + ", " + textBox2.Text + "] не будут учитываться при построении графика");
       }
}
```

Поскольку пользователь мог допустить ошибку и ввести точки, имеющие одинаковые значения X, но разные значения Y (что противоречит определению функции), необходимо добавить в программу механизм, позволяющий исключить из набора точек функции те, у которых имеются повторяющиеся значения X. Данный механизм реализован в настоящей программе следующим образом.

После получения введённых значений точек содержащий их список сортируется по значениям X с использованием имеющейся в языке C# функции сортировки *Sort*. Важно заметить, что в случае наличия в списке точек с одинаковыми координатами x их сортировка по y не осуществляется, таким образом, впоследствии из всех точек с координатами для интерполяции будет использоваться первая введённая такая точка.

```
listOfPoints.Sort((left, right) => left.x.CompareTo(right.x));
```

Далее в цикле перебираются элементы списка точек. Если значение x точки с индексом *i* равно значению точки с индексом *i-1*, пользователю выводится сообщение, информирующее о некорректности пользовательского ввода. Также будут выведены координаты точек, значения которых не будут учитываться при интерполяции функции. Такие точки будут удалены из списка

```
for (int i = 1, sizeOfTheList = listOfPoints.Count; i < sizeOfTheList; i++)
{
      if (listOfPoints[i].x == listOfPoints[i - 1].x)
      {
      MessageBox.Show("Введены несколько точек с одинаковыми координатами x! Точка
      с координатами [" + listOfPoints[i].x + "; " + listOfPoints[i].y "] не будет
      учитываться при интерполяции");
      listOfPoints.RemoveAt(i);
      --sizeOfTheList;
      --i;
      }
}
```

Для хранения интерполируемых точек создаётся список типа *Point*.

```
List<Point> listInterPolation = new List<Point>();
```

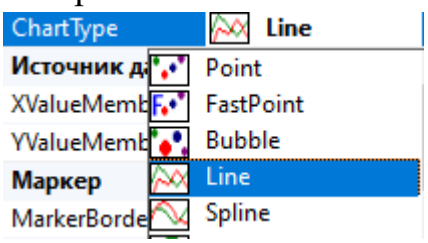
После чего запускается функция *Interpolation*, реализующая изложенный в конструкторском разделе алгоритм (к коду приведены комментарии).

```
void Interpolation()
{
      int n = listOfPoints.Count - 1; // количество интервалов между точками
      double[] k = new double[n + 1];
      double[] c = new double[n + 2];
      k[1] = 0;
      c[1] = 0;
      int i, j, l, m; // счётчики
      double a, b, q, g, h, d; // коэффициенты
      double x, y; // полученные при интерполяции координаты
      for (i = 2; i <= n; i++) // вычисление прогоночных коэффициентов/прямой ход
            {
            j = i - 1;
            m = j - 1;
            a = listOfPoints[i].x - listOfPoints[j].x;
            b = listOfPoints[j].x - listOfPoints[m].x;
            g = 2 * (a + b) - b * c[j];
            c[i] = a / g;
            k[i] = (3.0 * ((listOfPoints[i].y - listOfPoints[j].y) / a -
(listOfPoints[j].y - listOfPoints[m].y) / b) - b * k[j]) / g;
             }
      c[n + 1] = k[n];
      for (i = n; i >= 0; i--) // вычисление прогоночных коэффициентов/обратный ход
            {
            c[i] = k[i] - c[i] * c[i + 1];
            }

      int nx = 100; // количество интерполируемых точек между точками, заданными
пользователем
      for (i = 1; i <= n; i++) // интерполяция с использованием полученных
коэффициентов
            {
            h = (listOfPoints[i].x - listOfPoints[i - 1].x) / (nx - 1);
                  for (l = 0; l < nx; l++)
                  {
                  x = listOfPoints[i - 1].x + l * h;
                  q = listOfPoints[i].x - listOfPoints[i - 1].x;
                  g = x - listOfPoints[i - 1].x;
                  b = (listOfPoints[i].y - listOfPoints[i - 1].y) / q - (c[i + 1]
+ 2 * c[i]) * q / 3.0;
                  d = (c[i + 1] - c[i]) / q * g;
                  y = listOfPoints[i - 1].y + g * (b + g * (c[i] + d / 3.0));

                  p.x = Convert.ToDouble(x); // получение координаты x точки
                  p.y = Convert.ToDouble(y); // получение координаты y точки
                  listInterPolation.Add(p); // добавление точки в список
                   }
            }
}
```

После чего запускается модуль рисования полученного графика. Важно отметить, что возможна ситуация, при которой введённых точек будет недостаточно для интерполяции функции. Если точки не введены пользователем, либо введена только одна точка, список интерполируемых точек будет пуст. Очевидно, что в обоих случаях график функции нарисовать невозможно. Поэтому на этапе отображения графика используется перехват ошибок. Если построение графика функции невозможно, будет выведено сообщение, информирующее пользователя о недостаточном количестве введённых точек.

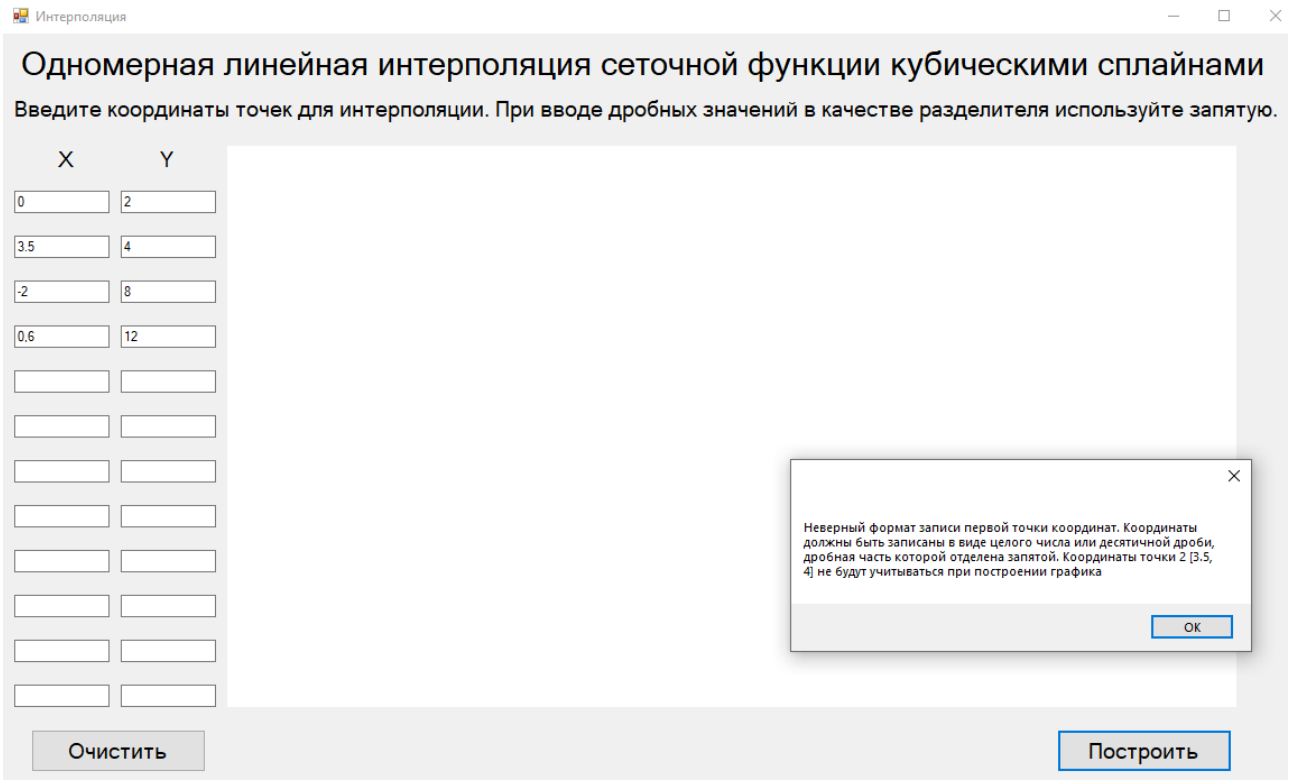
```
try
{
      Interpolation();
      chart1.ChartAreas[0].AxisX.Minimum = listInterPolation[0].x;
      chart1.ChartAreas[0].AxisX.Maximum =
                              listInterPolation[listInterPolation.Count - 1].x;
      this.chart1.Series[0].Points.Clear();

      foreach (Point point in listInterPolation)
            {
            this.chart1.Series[0].Points.AddXY(point.x, point.y);
             }
}

catch
{
      MessageBox.Show("Недостаточно точек для интерполяции функции");
}
```

Для удобства пользователя при вводе значений точек можно переключаться между текстовыми полями, используя клавишу Tab – приоритет переключения между ними настроен таким образом, что позволяет последовательно вводить координаты точек одну за другой, после чего пользователь переходит на кнопку «Построить».
Довольно интересной особенностью фреймворка Windows Forms является возможность построения графиков функций, используя уже имеющиеся средства построения сплайнов. Однако, для выполнения данной работы в соответствии с выданным заданием был использован тип графика «Line» - построение по точкам.



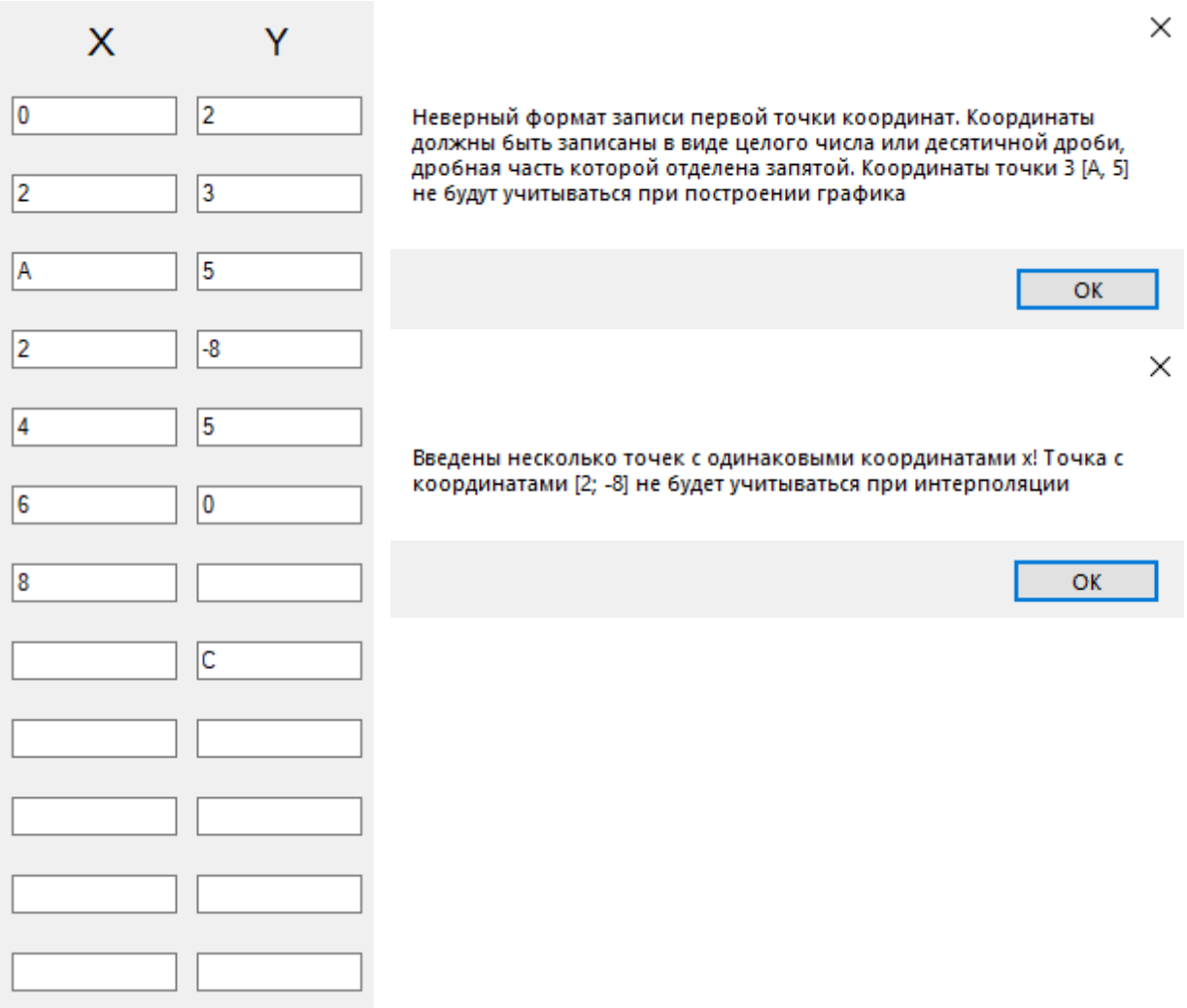
4. Результаты тестирования

Пример обработки некорректного ввода: пользователь добавил точку с координатами [3,5; 4], однако при вводе значения x вместо запятой в качестве разделителя использована точка. Пользователю выводится сообщение,

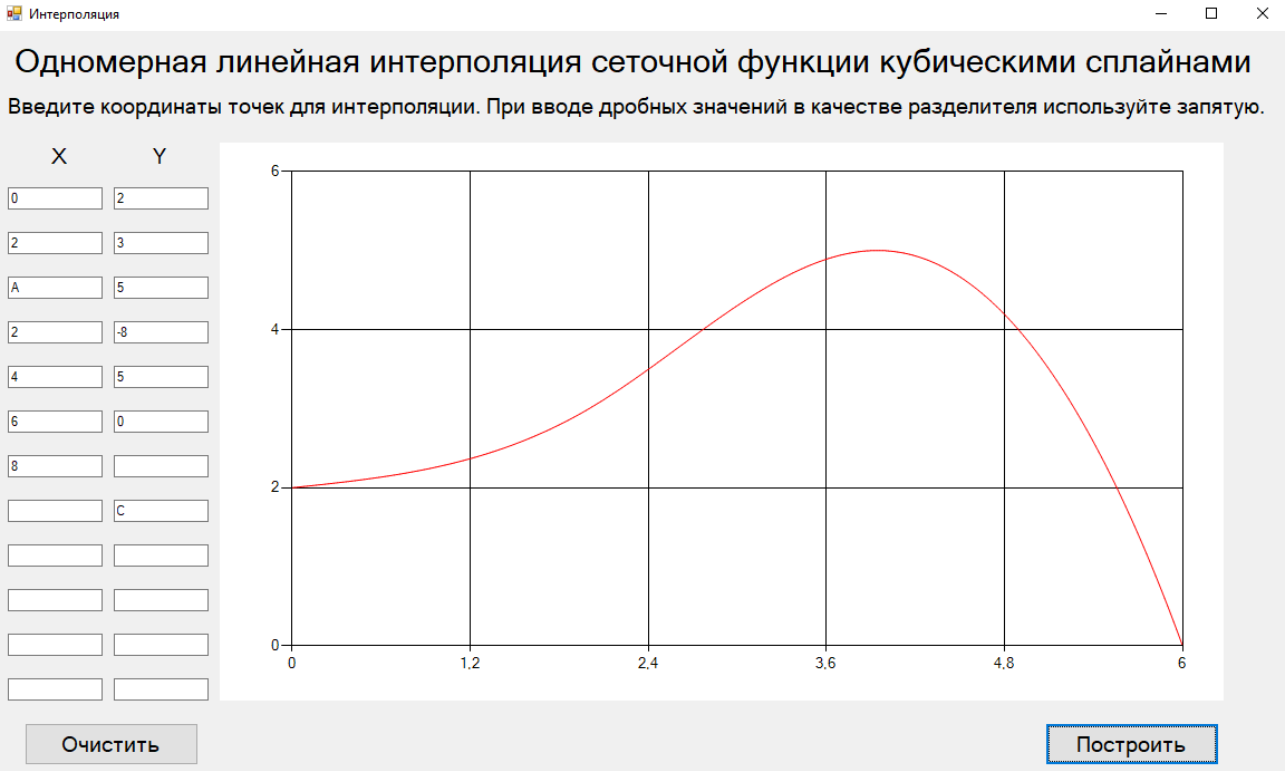


после чего осуществляется построение графика функции без учёта этой точки.

Ещё один пример некорректного пользовательского ввода на следующем наборе значений: пользователю поочерёдно выводятся следующие сообщения:



После чего программа строит график по корректно заданным точкам:



В качестве проверки работы программы мы можем построить график какой-либо известной функции, например, y = sin (x) на интервале [-5; 5]. Используя известные значения данной функции в определённых точек, попробуем осуществить её интерполяцию, после чего увидим, будет ли совпадать полученный нами график с графиком функции y = sin (x).

Будем использовать таблицу значений данной функции ниже с точностью до двух знаков после запятой – в образовательных целях такой точности будет вполне достаточно, чтобы получить представление о поведении функции и виде её графика.

X	-5	-4	-3	-2	-1	0	1	2	3	4	5
Y	0.95	0.75	-0.14	-0.90	-0.84	0	0.84	0.90	0.14	-0.75	-0.95

Очевидно, что получившийся график функции полностью повторяет поведение функции y = sin (x) – мы видим периодическую нечётную функцию, значение которой лежит между -1 и 1.

Заключение

В ходе выполнения научно-исследовательской работы был изучен алгоритм одномерной линейной интерполяции сеточной функции кубическими сплайнами и разработана программа, которая позволяет осуществить такую интерполяцию, используя заданные пользователем точками и вывести график такой функции. Интерфейс программы учитывает возможные ошибки, которые может допустить пользователь и предупреждает об их появлении.

Список использованных источников информации

1. Градов В.М. Компьютерные технологии в практике математического моделирования: Учебное пособие. – Ч.1. – М.: Изд-во МГТУ им. Баумана, 2005. – 108 с.

2. Градов В.М. Лекционные материалы по дисциплине «Моделирование».

3. Документация по C#. [Электронный ресурс].
URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/