Е.А. Каминский, маг.; рук. А.И. Якимов, к.т.н., доц.
(Белорусско-Российский университет, г. Могилев, Беларусь)

ПРИЛОЖЕНИЕ ДЛЯ КОМПЛЕКСНОГО АНАЛИЗА КОЛИЧЕСТВЕННЫХ ДАННЫХ В ПРОИЗВОДСТВЕННОМ ПРОЦЕССЕ

Обработка и анализ данных на производстве является одним из наиболее важных направлений развития современных компьютерных производственных систем. Скорость и качество обработки данных напрямую зависит от развития программных средств и технологий сбора, передачи и представления информации, используемых в них. Программные системы, с помощью которых в настоящее время решаются задачи обработки и анализа специализированных данных (MS Excel, Statistica и др) [1], не всегда соответствуют современным технологическим требованиям. Класс систем, обеспечивающих подобные обработку и анализ данных, не всегда предоставляют удобные средства визуализации и хранения данных. Исходя из выше сказанного, важность задачи по разработке специализированного приложения для анализа производственных данных становится очевидной. Такое приложение сможет работать с различными наборами входных данных, осуществлять отображение результатов принятия решения, осуществлять связь с базами данных для отображения тенденций в производственной деятельности.

Для сохранения данных о результатах деятельности используется СУБД Microsoft SQL Server. Выбор обусловлен наличием функциональности, требуемой для решения задач приложения, а также интеграцией с программными средствами разработки приложения, что заметно упрощает создание связей между базой данных и приложением.

Ниже приведен участок кода, отвечающий за сохранение данных о процессе в базу данных. Для сохранения предлагается вызов специально созданной хранимой в базе данных процедуры, код которой также представлен ниже:

```
SqlCommand cmd2 = new SqlCommand("addcase", connection); //создание SQL-команды;
cmd2.CommandType = CommandType.StoredProcedure;          //определение типа команды;
cmd2.Parameters.AddWithValue("@process", process);       //передача
cmd2.Parameters.AddWithValue("@id", processid);          //параметров
cmd2.Parameters.AddWithValue("@xval", xval);             //в
cmd2.Parameters.AddWithValue("@yval", yval);             //хранимую
cmd2.Parameters.AddWithValue("@data", DateTime.Now);     //процедуру;
cmd2.ExecuteNonQuery();                                  //выполнение процедуры
```

```
CREATE procedure addcase
@process  nvarchar(150), @id int,@xval nvarchar(max), @yval nvarchar(max), @data datetime
As
insert into dbo.Данные
values (@process,@id, @number, @xval,@yval, @data).
```

Непосредственно принятие решений формируется на основе ЕСЛИ ТО правил, используемых в математической логике. Правила записываются в виде ЕСЛИ *x* это *A*, ТО *y* это *B*. В данном случае *x* – это некоторый аналитический

параметр, A – его значение, y – значение отклонения от нормы, B – факт наличия отклонения (истина или ложь). Такие высказывания формируют базу правил, позволяющую сформировать целостную картину по выбранному случаю.

Программная реализация правил осуществляется при помощи условных блоков *if-else*, повторяющих структуру правила. Ниже представлен фрагмент кода с реализацией некоторых правил:

```
string conc = "";
if (right <= MaxVal * 1.2 && right >= MaxVal * 0.8 && bezier1[indexMax + 2] <= MaxVal * 1.2 && bezi-
er1[indexMax + 2] >= MaxVal * 0.8)
            conc = conc + "Вывод1. ";
        for (int i = indexMax + 1; i < bezier1.Length - 1; i++)  //цикл для нахождения резких подъёмов
        {                                                        // и падений графика
            if (bezier1[i] >= MaxVal * 0.6)
                k++;
            if (bezier1[i] <= MaxVal * 0.1)
                l++;
            if (bezier1[i] < bezier1[i - 1])
                m++;
        }
        if (k >= 2 && l >= 1)                              //правило для анализа графика, исходя из числа
            conc = conc + "Вывод2. ";                          //пиков и падений;
        if (bezier1.Length / indexMax >= 4 && m == bezier1.Length - indexMax - 2)  //если глобальный
            conc = conc + "Вывод3. ";                          //максимум в первой четверти графика;
        if (MaxVal >= 40)                                  //правило для проверки максимального
            conc = conc + "Вывод4. ";                          //значения на превышение порога;
        if (MaxVal < 15)                                   //правило для проверки максимального
            conc = conc + "Вывод5. ";                     //значения на соответствие минимальному порогу;
        if (RB2.IsChecked == true && MaxVal < 20)
            conc = conc + "Вывод6. ";
        if (avg < 10)                                      //правило для проверки среднего значения;
            conc = conc + "Вывод7.";
        if (conc1 == conc)                                 //если никакие из предыдущих правил не
            conc = conc + "Вывод8.";                       //принимают значение истина
        return conc;
```

Несмотря на то, что система принимает некоторое решение, оно носит лишь рекомендательный характер. Последнее слово в любом случае остается за человеком, и для облегчения принятия этого решения необходимо реализовать аппарат, позволяющий представить введенные данные в виде графика. Так как данные вводятся дискретно, существует необходимость создания механизма аппроксимации, который позволит представить набор точек в виде единой непрерывной линии. Одним из вариантов решения этой задачи является применение кривых Безье [2].

Алгоритм кривых Безье основывается на уравнениях параметрических кривых. При разработке алгоритма было высказано предположение, что для построения аппроксимирующей кривой для n точек достаточно будет использовать уравнения со второй и третьей старшей степенью, использующих соответственно три и четыре опорные точки. Функция $B(t)$, описывающая кривую Безье на заданном интервале, соответственно со второй и третьей степенью имеет следующий вид:

$$B(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2, t \in [0,1]; \quad (1)$$

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, t \in [0,1], \quad (2)$$

где P_0, P_1, P_2, P_3 – опорные точки для построения кривой, t – параметр, изменяющийся от 0 до 1.

В ходе тестирования было выявлено, что использование шага 0,05 для параметра t обеспечивает необходимую гладкость итоговой кривой. Для построения программа будет брать 4 точки и строить кубические кривые до тех пор, пока не останется 3 точки или менее (с учётом точки из предыдущей итерации). Тогда возможны следующие варианты: осталось три точки (в этом случае вместо кубической кривой используется квадратная; осталась две точки (на предыдущем этапе вместо одной кубической кривой используют две квадратные); осталась одна точка (используются только кубические кривые).

Ниже представлена часть кода реализации алгоритма построения кривых Безье:

```
    public double y;
    public double x;
    public static double t, t2, t3, nt, nt2, nt3;
    Bezier() { x = y = 0.0; }
    Bezier(int _x, double _y) { x = _x; y = _y; }
    public static double[,] Cubic(double[,] outArray, int[] xData, double[] yData, int k, int stoppoint)//куби-
    {                                                                                                  //ческая;
       double[,] points = new double[4, 2];  //массив для каждых отдельно взятых четырех точек;
       for (int m = 0; m < 4; m++)          //цикл для формирования мини-массива из четырех точек для
       {                                       //формирования отдельного сплайна;
          points[m, 0] = xData[stoppoint + m];
          points[m, 1] = yData[stoppoint + m];
       }
       for (t = 0; t <= 1.05; t += 0.05)       //цикл для перебора параметра t;
       {
          t2 = t * t;            //коэффициенты
          t3 = t2 * t;           //для построения
          nt = 1.0 - t;          //кубической
          nt2 = nt * nt;         //кривой
          nt3 = nt2 * nt;        //Безье;
          outArray[k, 0] = nt3 * points[0, 0] + 3.0 * t * nt2 * points[1, 0] + 3.0 * t2 * nt * points[2, 0] + t3
* points[3, 0];                      //формирование выходного массива x;
          outArray[k, 1] = nt3 * points[0, 1] + 3.0 * t * nt2 * points[1, 1] + 3.0 * t2 * nt * points[2, 1] + t3
* points[3, 1];                      //формирование выходного массива y;
          k++;
       }
       return outArray;      //возврат массива;
    }
    public static double[,] Square(double[,] outArray, int[] xData, double[] yData, int k, int stoppoint)//квад-
    {                                                                                                  //ратичная;
       double[,] points = new double[3, 2];   //цикл для формирования мини-массива из трех точек для
       for (int m = 0; m < 3; m++)             //формирования отдельного сплайна;
       {
          points[m, 0] = xData[stoppoint + m];
          points[m, 1] = yData[stoppoint + m];
```

```
    }
    for (t = 0; t <= 1.05; t += 0.05)        //цикл для перебора параметра t;
    {
       t2 = t * t;     //коэффициенты
       t3 = t2 * t;   //для построения
       nt = 1.0 - t;  //квадратичной
       nt2 = nt * nt;//кривой Безье
       outArray[k, 0] = nt2 * points[0, 0] + 2.0 * t * nt * points[1, 0] + t2 * points[2, 0];   //массив x;
       outArray[k, 1] = nt2 * points[0, 1] + 2.0 * t * nt * points[1, 1] + t2 * points[2, 1];   //массив y;
       k++;
    }
    return outArray;                    //возврат массива;
}.
```

Для тестирования приложения (рисунок 1) использовался компьютер с процессором Intel Core i3-4170 и 8 ГБ ОЗУ под управлением операционной системы Windows 7.

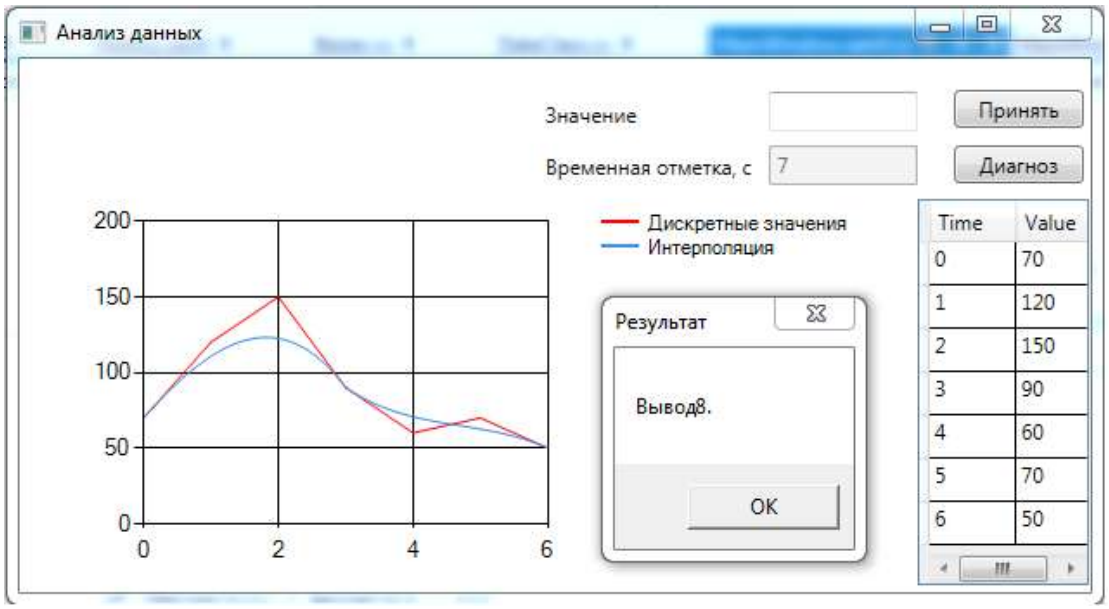

Рисунок 1 – Результат работы программного приложения

Таким образом, представлено новое приложение, выполняющее анализ количественных производственных данных и представляющее по ним аналитические выводы. Применение его в производстве позволит уменьшить время анализа производственных данных, улучшить качество принимаемых решений, свести вероятность неверного решения к минимуму.

Литература

1. Рубан А.И., Кузнецов А.В. Методы обработки экспериментальных данных. Красноярск, 2008. – 80 с. [Электронный ресурс]. – Режим доступа: http://ikit.edu.sfu-kras.ru/files/17/lab/lab.pdf.

2. Кантор И. Кривые Безье. 2018. URL: https://learn.javascript.ru/bezier (дата обращения: 18.01.2018).