ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

ИСТРУМЕНТАЛЬНОЕ СРЕДСТВО ИНТЕРПОЛИРОВАНИЯ ФУНКЦИЙ

Выпускная квалификационная работа
обучающейся по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения, группы 07001402
Рыгиной Кристины Григорьевны

Научный руководитель
к.т.н.,доцент Румбешт В.В.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В реальной жизни очень часто приходится решать задачи интерполяции функций. Решение данной задачи весьма востребовано.

Интерполяция применяется в различных областях, например:

- интерполяция поверхности осадков в метеорологии;
- интерполяция поверхности высот в геологии;
- интерполяция поверхности распределения в медицине;
- вычисление уклонов и экспозиции склонов в строительстве;
- проведение баллистической экспертизы в криминалистике.

Существует множество методов решения этой задачи, которые отличают точностью, видом графика и так далее. Причем не существует идеального метода. Существует множество методов и выбор метода зависит от специфики решаемой конкретной задачи, поэтому, возникает необходимость в создании инструментального средства, которое реализует как можно больше существующих методов, предоставляющий пользователю простой и удобный интерфейс для решения задачи. На данный момент уже есть определенные разработки в области программных средств такого типа, но все они не являются законченными, или вовсе закрыты. Поэтому создание приложения, которое будет обладать функционалом, пусть и узкоспециализированным, но полным в этой области, довольно актуально.

Для решения задач интерполяции разработан ряд инструментальных средств, таких как MathCAD, MATLAB, – универсальные математические пакеты, которые поддерживают и другие методы. Чтобы ими пользоваться нужно иметь представления о методах, способах ведения. Например, в MATLAB приходится самостоятельно писать формулы с использованием встроенного языка. Поэтому становится актуальным создание

инструментального средства, которым сможет пользоваться специалист своей предметной области.

Целью работы является разработка инструментального средства, которое будет обладать способностью интерполирования функции различными видами интерполяции, а также, - построения графиков на основании заданного массива точек.

Задачи:

1. Провести обзор существующих методов решения задач интерполяции.
2. Провести обзор существующих инструментальных средств и программ для решения задач интерполяции.
3. Сформулировать требования к инструментальному средству.
4. Реализовать алгоритмы интерполяции.
5. Разработать интерфейс инструментального средства.
6. Разработать программное обеспечение.
7. Провести испытания программного обеспечения.

При разработке должны быть учтены следующие требования: малая стоимость приложения или его бесплатность; дружелюбный для пользователя интерфейс; возможность легко освоить интерфейс программы; минимальные технические требования, позволяющие запустить приложение на устаревших ПК. Также, программа должна оперативно выполнять вычисления, выбранные пользователем, информировать о неправильно введенных данных или исправлять их.

В первой главе ВКР приведены теоретические сведения о методах решения задач интерполирования функций, также представлены требования к инструментальному средству.

Во второй главе происходит реализация алгоритмов интерполяции и разработка интерфейса приложения.

В третьей главе проводится тестирование инструментального средства.

Выпускная квалификационная работа состоит из 58 страниц, 35 рисунков, 11 таблиц и приложения включающего 15 страниц.

1. ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА ИНТЕРПОЛИРОВАНИЯ ФУНКЦИЙ

1.1 Задача интерполяции и методы её решения

Задача интерполяции

Пусть функция f(x) задана набором точек (x_i, y_i) на интервале $[a, b]$:

$$y_i = f(x_i),\ i = 0,1 \dots n\ ,\ a \le x_i \le b \qquad (1.1)$$

Задача интерполяции – найти функцию $F(x)$, принимающую в точках x_i те же значения y_i. Тогда, *условие интерполяции*:

$$F(x_i) = y_i \qquad (1.2)$$

При этом предполагается, что среди значений x_i нет одинаковых. Точки x_i называют *узлами интерполяции*.

Если $F(x)$ можно найти только на отрезке $[a, b]$ – то это задача *интерполяции*, а если за пределами первоначального отрезка, то это *экстраполяции*.

При нахождении интерполяционной функции $F(x)$ задача имеет множество решений, так как через заданные точки (x_i, y_i) можно провести бесконечно много кривых, каждая из которых будет графиком функции, для которой выполнены все условия интерполяции. Для решения практических задач важен случай интерполяции функции многочленами:

$$F(x) = P_m(x_i) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_m \cdot x^m,\ i = 0,1 \dots m \qquad (1.3)$$

При этом искомый полином называется *интерполяционным полиномом*.

При построении одного многочлена для всего рассматриваемого интервала $[a, b]$ для нахождения коэффициентов многочлена необходимо решить систему уравнений, построенную на основе полинома (1.3). Данная система содержит $n + 1$ уравнение, следовательно, с ее помощью можно определить $n + 1$ коэффициент. Поэтому максимальная степень интерполяционного многочлена $m = n$, и многочлен принимает вид:

$$P_n(x_i) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n, i = 0,1 \ldots n \qquad (1.4)$$

Глобальная интерполяция

Если задан $n + 1$ узел интерполяции, то на этих узлах можно построить один интерполяционный многочлен n-й степени, $n - 1$ многочленов первой степени и большой набор многочленов степени меньше n, опирающиеся на некоторые из этих узлов.

Теоретически максимальную точность обеспечивает многочлен более высокой степени. Однако на практике наиболее часто используют многочлены невысоких степеней, во избежание погрешностей расчета коэффициентов при больших степенях многочлена.

Если функция $f(x)$ интерполируется на отрезке $[a, b]$ с помощью единого многочлена $P_m(x)$ для всего отрезка, то такую интерполяцию называют *глобальной* [1].

Кусочно-линейная интерполяция

Простейшим и часто используемым видом локальной интерполяции является линейная (или кусочно-линейная) интерполяция. Она заключается в том, что узловые точки соединяются отрезками прямых (рис.1.1), то есть через каждые две точки (x_i, y_i) и (x_{i+1}, y_{i+1}) проводится прямая, то есть составляется полином первой степени:

$$F(x) = a_0 + a_1 \cdot x, \text{ при } x_{i-1} \le x \le x_i \qquad (1.5)$$

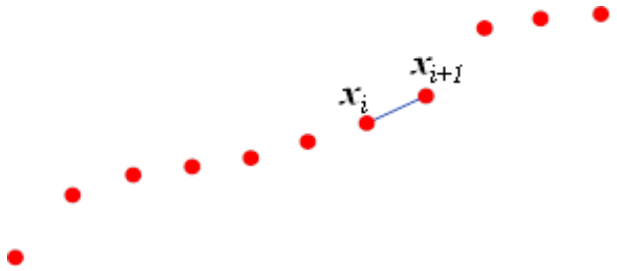

Рис. 1.1. Кусочно-линейная интерполяция

Коэффициенты a_0 и a_1 разные на каждом интервале $[x_i, x_{i+1}]$, и находятся из выполнения условий интерполяции на концах отрезка:

$$\begin{cases} f_{i-1} = a_0 + c_1 \cdot x_{i-1} \\ f_i = a_0 + a_1 \cdot x_i \end{cases} \quad (1.6)$$

Из системы уравнений (1.6) можно найти коэффициенты:

$$a_0 = f(x_{i-1}) - a_1 \cdot x_{i-1},\ a_1 = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \quad (1.7)$$

При использовании кусочно-линейной интерполяции сначала нужно определить интервал, в который попадает значение *x*, а затем подставить его в выражение (1.5), используя коэффициенты для данного интервала [1-5].

Кусочно-квадратичная интерполяция

В случае квадратичной интерполяции (рис. 1.1), для каждых трех узловых точек (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}), строится уравнение параболы [5, 8, 23]:

$$F(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2, \text{ при } x_{i-1} \le x \le x_{i+1} \quad (1.8)$$

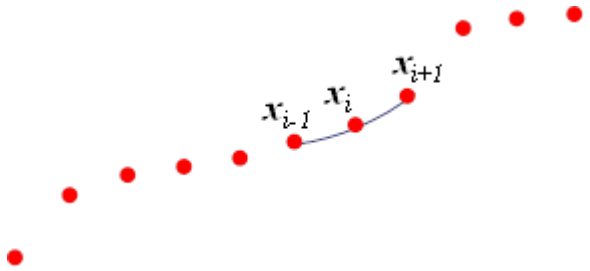

Рис. 1.2. Кусочно-квадратичная интерполяция

Здесь коэффициенты a_0 , a_1 и a_2 разные на каждом интервале $[x_{i-1}, x_{i+1}]$ и определяются решением системы уравнений для условия прохождения параболы через три точки:

$$\begin{cases} f_{i-1} = a_0 + a_1 * x_{i-1} + a_2 * x_{x-i}^2 \\ f_i = a_0 + a_1 * x_i + a_2 * x_x^2 \\ f_{i+1} = a_0 + a_1 * x_{i+1} + a_2 * x_{x+i}^2 \end{cases} \quad (1.9)$$

Из системы уравнений (1.9) можно найти коэффициенты:

$$a_0 = f(x_{i-1}) - a_1 \cdot x_{i-1} - a_2 \cdot x_{i-1}^2$$

$$a_1 = \frac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}} - a_2 * (x_i + x_{i-1}) \quad (1.10)$$

$$a_{2=} \frac{f(x_{i+1})-f(x_{i-1})}{(x_{i+1}-x_{i-1})*(x_{i+1}-x_i)} - \frac{f(x_i)-f(x_{i-1})}{(x_i-x_{i-1})*(x_{i+1}-x_i)}$$

Многочлен Лагранжа

При глобальной интерполяции на всем интервале $[a,b]$ строится единый многочлен. Одной из форм записи интерполяционного многочлена для глобальной интерполяции является многочлен Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} y_i * l_i(x) \quad (1.11)$$

где $l_i(x)$ – базисные многочлены степени n:

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x-x_j}{x_i-x_j} = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \quad (1.12)$$

То есть многочлен Лагранжа можно записать в виде:

$$L_n(x) = \sum_{i=0}^{n} y_i * \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x-x_j}{x_i-x_j} \quad (1.13)$$

Многочлен $l_i(x)$ удовлетворяет условию $l_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$. Это условие означает, что многочлен равен нулю при каждом x_j кроме x_i, то есть $x_0, x_1 \ldots x_{i-1}, x_{i+1}, \ldots x_n$ – корни этого многочлена. Таким образом, степень многочлена $L_n(x)$ равна n и при $x \neq x_i$ обращаются в ноль все слагаемые суммы, кроме слагаемого с номером $i = j$, равного y_i.

Выражение (1.11) применимо как для равноотстоящих, так и для не равноотстоящих узлов. Погрешность интерполяции методом Лагранжа зависит от свойств функции *f(x)*, от расположения узлов интерполяции и точки *x*. Полином Лагранжа имеет малую погрешность при небольших значениях *n* (*n*<20). При больших *n* погрешность начинает расти, что свидетельствует о том, что метод Лагранжа не сходится (то есть его погрешность не убывает с ростом *n*).

Многочлен Лагранжа в явном виде содержит значения функций в узлах интерполяции, поэтому он удобен, когда значения функций меняются, а узлы интерполяции неизменны. Число арифметических операции, необходимых для построения многочлена Лагранжа, пропорционально n^2 и является наименьшим для всех форм записи. К недостаткам этой формы записи можно отнести то, что с изменением числа узлов приходится все вычисления проводить заново.

Кусочно-линейная и кусочно-квадратичная локальные интерполяции являются частными случаями интерполяции многочленом Лагранжа [2-6, 13].

Многочлен Ньютона

Другая форма записи интерполяционного многочлена – интерполяционный многочлен Ньютона с разделенными разностями. Пусть функция *f(x)* задана с произвольным шагом, и точки таблицы значений пронумерованы в произвольном порядке [1-6, 11].

Разделенные разности нулевого порядка совпадают со значениями функции в узлах. Разделенные разности первого порядка определяются через разделенные разности нулевого порядка:

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \qquad (1.14)$$

Разделенные разности второго порядка определяются через разделенные разности первого порядка:

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i} \qquad (1.15)$$

Разделенные разности *k*-го порядка определяются через разделенные разности порядка *k-1*:

$$f(x_i, x_{i+1}, \dots x_{i+k}) = \frac{f(x_{i+1}, \dots x_{i+k}) - f(x_i, \dots x_{i+k+1})}{x_{i+k} - x_i} \qquad (1.16)$$

Используя понятие разделенной разности интерполяционный многочлен Ньютона можно записать в следующем виде:

$$P_n(x) = f(x_0) + f(x_0, x_1) * (x - x_0) + f(x_0, x_1, x_2) * (x - x_0) * (x - x_1) + \dots + f(x_0, x_1, \dots, x_n) * (x - x_0) * (x - x_1) \dots * (x - x_{n-1}) \qquad (1.17)$$

За точностью расчета можно следить по убыванию членов суммы (1.17). Если функция достаточно гладкая, то справедливо приближенное

равенство $f(x) - P_n(x) \approx P_{n+1}(x) - P_n(x)$. Это приближенное равенство можно использовать для практической оценки погрешности интерполяции: $\varepsilon_n = |P_{n+1}(x) - P_n(x)|$.

Интерполяционная формула Гауса – формула, использующая в качестве узлов интерполяции ближайшие к точке интерполирования *x* узлы. Если $x = x_0 + th$, то формула

$$G_{2n+1}(x_0 + th) = f_0 + f_{\frac{1}{2}}^1 t + f_0^2 \frac{t(t-1)}{2!} + \cdots$$
$$+ f_0^{2n} \frac{t(t^2-1)\ldots[t^2-(n-1)^2](t+n)}{(2n)!} \qquad (1.18)$$

Написанная по узлам $x_0, x_0 - h, x_0 + h, \ldots, x_0 - nh, x_0 + nh,$ называется формулой Гаусса для интерполирования назад. В формулах (1.18) и (1.19) использованы конечные разности, определяемые следующим образом:

$$f_{i+1/2}^1 = f_{i+1} - f_i, f_i^m = f_{i+\frac{1}{2}}^{m-1} - f_{i-\frac{1}{2}}^{m-1} \qquad (1.19)$$

Преимущество интерполяционной формулы Гаусса состоит в том, что указанный выбор узлов интерполяции обеспечивает наилучшую оценку остаточного члена по сравнению с любым другим выбором, а упорядоченность узлов по мере их близости к точке интерполяции уменьшает вычислительную погрешность интерполирования.

В геологии проводится опробование месторождение и определяется концентрация полезных ископаемых в определенных точках, с помощью интерполяции можно оценить концентрацию в промежуточных точках. Список реальных примеров легко продолжить.

Более формально, пусть нам даны значения некоторой функции в некоторых точках области определения [4, 11, 22].

Перед нами стоит задача наиболее точно определить вид этой функции по заданным значениям. Один из возможных подходов - прибегнуть к интерполяции сплайнами.

Сплайн – функция, которая вместе с несколькими производными непрерывна на всем заданном отрезке [a,b], а на каждом частичном отрезке $[x_i,\ x_{i+1}]$ в отдельности является некоторым алгебраическим многочленом.

Степенью сплайна называется максимальная по всем частичным отрезкам степень многочленов, а дефектом сплайна - разность между степенью сплайна и порядком наивысшей непрерывной на [a,b] производной. Например, непрерывная ломанная является сплайном степени 1 с дефектом 1 (так как сама функция – непрерывна, а первая производная уже разрывна).

На практике наиболее часто используются *кубические* сплайны $S_3(x)$ - сплайны третьей степени с непрерывной, по крайней мере, первой производной. При этом величина $m_i = S_3'$, называется наклоном сплайна в точке (узле) x_i.

Разобьём отрезок [a,b] на N равных отрезков $[x_i,\ x_{i+1}]$, где $x_i = a + ih$, $i=0,1,\ldots,N-1$, $x_n = b, h = \frac{b-a}{N}$.

Если в узлах x_i, x_{i+1} заданы значения f_i, f_{i+1}, которые принимает кубический сплайн, то на частичном отрезке $[x_i,\ x_{i+1}]$ он принимает вид:

$$S_3(x) = \frac{(x_{i+1}-x)^2(2(x-x_i)+h)}{h^3} f_i + \frac{(x-x_i)^2(2(x_{i+1}-x)+h)}{h^3} f_{i+1} + \\ + \frac{(x_{i+1}-x)^2(x-x_i)}{h^2} m_i + \frac{(x-x_i)^2(x-x_{i+1})}{h^2} m_{i+1}. \qquad (1.20)$$

В самом деле, это легко проверить, рассчитав $S_3(x)$ и $S_3'(x)$ в точках x_i, x_{i+1}.

Можно доказать, что если многочлен третьей степени принимает в точках x_i, x_{i+1} значения f_i, f_{i+1} и имеет в этих точках производные, соответственно, m_i, m_{i+1}, то он совпадает с многочленом (1.19).

Таким образом, для того, чтобы задать кубический сплайн на отрезке, необходимо задать значения f_i, m_i, i=0,1…,N в N+1 в узле x_i.

Кубический сплайн, принимающий в узлах те же значения f_i, что и некоторая функция, называется *интерполяционным* и служит для аппроксимации функции f на отрезке [a,b] вместе с несколькими производными.

Пусть $S_3''(x_i + 0)$ - значение $S_3''(x)$ в узле x_i справа, его мы найдем из выражения (1.18), а $S_3''(x_i - 0)$ - значение $S_3''(x)$ в узле x_i слева – оно находится из соответствующего выражения $S_3(x)$ на частичном отрезке $[x_i, x_{i+}]$, которое получается из (1.18) заменой i на i-1.

Тогда получим:

$$S_3''(x_i + 0) = -\frac{4m_i}{h} - \frac{2m_{i+1}}{h} + 6\frac{f_{i+1}+f_i}{h^2} \quad (1.21)$$

$$S_3''(x_i - 0) = \frac{2m_{i-1}}{h} + \frac{4m_i}{h} - 6\frac{f_i - f_{i-1}}{h^2} \quad (1.22)$$

Потребуем непрерывность $S_3''(x)$ в узлах:

$$S_3''(x_i - 0) = S_3''(x_i - 0), \text{ i=1,2,…, N-1.}$$

Тогда получим систему линейных алгебраических уравнений относительно наклонов:

$$m_{i-1} + 4m_i + m_{i+1} = \frac{3(f_{i+1} - f_{i-1})}{h}, i = 1,2, \dots, N-1. \quad (1.23)$$

Так как система содержит N+1 неизвестных, необходимо задать два дополнительных условия, называемые *краевыми.*

Приведем три варианта задания краевых условий:

1) В случае, когда известны $m_N = f_N$ задаем

$m_0 = f_0', m_N = f_N'$.

2) Производные f_0' , f_N' аппроксимируем формулами численного дифференцирования третьего порядка точности:

$$m_0 = \frac{1}{6h}(-11f_0 + 18f_1 - 9f_2 + 2f_3), \quad (1.24)$$

$$m_N = \frac{1}{6h}(11f_N - 18f_{N-1} + 9f_{N-2} - 2f_{N-3}). \quad (1.25)$$

3) Иногда бывают известны значения f'' на концах отрезка [a,b], т.е. величины $f_0'' = f''(a), f_N'' = f''(b)$. Тогда требования $S_3''(a) = f_0'', S_3''(b) = f_N''$ приводят к краевым условиям:

$$m_0 = \frac{-m_1}{2} - \frac{3}{2}\frac{f_1 - f_0}{h} - \frac{h}{4}f_0'',$$

$$m_N = \frac{-m_{N-1}}{2} + \frac{3}{2}\frac{f_N - f_{N-1}}{h} + \frac{h}{4}f_N'' \ . \quad (1.26)$$

Условия (1.24)-(1.25) можно комбинировать, то есть выбирать их независимо в левом и правом узлах.

Система (1.26) при всех рассмотренных краевых условиях имеет единственное решение, которое можно найти с помощью методов прогонки и итераций.

Таким образом, решая систему (1.26) при выбранных краевых условиях, находим наклоны m_i, i=0,1,…,N, во всех узлах. Затем по формуле (1.25) задаем сплайн на каждом частичном отрезке [x_i, x_{i+1}], i=0,1,…,N-1. Построенный данным глобальным способом сплайн $S_3(x)$ имеет дефект не больше единицы, так как этот сплайн обладает на отрезке [a,b] непрерывной второй производной $S_3''(x)$.

Кубический эрмитов сплайн — сплайн, построенный из кубических полиномов с использованием эрмитовой интерполяции, в соответствии с которой интерполируемая функция задается не только своими значениями в *n* точках, но и её первыми производными [11, 22-23]. Для заданной

интерполяционной сетки x_k для *k=1,…,n*, и заданного значения независимой переменной *x* вычисление функции проводится в соответствующем интервале с известными граничными значениями функции p и ее производной m. Для упрощения вычислений делается замена независимой переменной *x* на независимую переменную t по формуле $t = (x - x_k)/(x_{k+1} - x_k)$. В результате такой замены левая граница интервала становится равной *0*, а правая *1*. Кубический полином, служащий для вычисления интерполируемой функции в соответствующем интервале, имеет вид:

$$p(t) = (2t^3 - 3t^2 + 1)p_k + (t^3 - 2t^2 + t)(x_{k+1} - x_k)m_k + \\ +(-2t^3 - 3t^2)p_{k+1} + (t^3 - t^2)(x_{k+1} - x_k)m_{k+1} \qquad (1.27)$$

В вышеприведенной формуле (1.27) значения относятся к независимой переменной t. Для вычисления необходимо исходный значения производных умножить на длины интервалов $(x_{k+1} - x_k)$. Как следует из формулы, значение интерполируемой функции вычисляется с помощью четырех кубических полиномов $h_{00}(t), h_{10}(t), h_{01}(t), h_{11}(t)$. Эти полиномы отнюдь не являются классическими полиномами Эрмита. На практике обычно известны лишь значения функции в узловых точках, но не значение первой производной. Для вычисления значений первой производной используются различные способы. Простейшим является вычисление среднего арифметического значения разделенных первых разностей на двух соседних интервалах.

$$m_k = \frac{p_{k+1} - p_k}{2(t_{k+1} - t_k)} + \frac{p_k - p_{k-1}}{2(t_k - t_{k-1})} \qquad (1.28)$$

В так называемом кардинальном сплайне используется формула:

$$m_k = (1 - c)\frac{p_{k+1} - p_{k-1}}{t_{k+1} - t_{k-1}} \qquad (1.29)$$

В этой формуле параметр *c* изменяется от 0 до 1. В соответствии с этой формулой производная в середине отрезка равняется разделенной первой разности на всем отрезке, умноженной на некий коэффициент.

1.2 Обзор существующих программных средств

В наше время существует не так много уже готовых продуктов (программных средств), выполняющих поставленную задачу, и большинство из них имеют определенные недостатки, в большей степени связанные с ценой продукта. Если и есть бесплатные продукты, то они отличаются слабым функционалом. Ниже представлена таблица сравнения приложений.

Таблица 1.1

Сравнение готовых продуктов

	Дружелюбный интерфейс	Богатый функционал	Доступность в цене	Работает на слабых ПК	Легко освоить
MatLab	Нет	Да	2150$	Нет	Нет
MathCad	Да	Да	от 22 250р	Да	Нет
Mathematica	Да	Да	от 155$	Нет	Нет
Derive (Проект закрыт)	Да	Да	Неизвестно	Да	-
Scilab	Нет	Да	Бесплатно	Да, но не рекомендуется	Нет

Нельзя сказать, что продукты, описанные в таблице 1.1 - чем-либо плохи. Они обладают богатым функционалом, который можно получить лишь с годами разработки большой командой; некоторые из них обладают техническими характеристиками, которые позволяют запустить продукт на

слабом ПК, но за это необходимо заплатить, - как, в буквальном смысле, большой стоимостью, так и дружелюбностью интерфейса. Единственный «общий» минус - пользователям, которые раньше не работали в данном приложении, будет очень тяжело освоить его.

Исходя из этого, была поставлена задача разработки узкоспециализированного программного средства, позволяющего интерполировать введенные пользователем функции. Основным преимуществом должен быть «минимализм» вкупе с функциональностью, т.е. пользователь может делать выбор того, что хочет; вводить / загружать / сохранять данные, которые используются для расчетов, и не более, что, соответственно, повлияет на возможность легче освоить приложение. В рамках узкой специализации, а конкретно, - интерполирования функций, - приложение должно отличаться богатым функционалом и общедоступностью.

1.3 Требования к инструментальному средству

Функциональное назначение.

Программа «Интерполяция функций» (далее Программа) предназначена для:

- возможность интерполирования следующими методами: линейная, квадратичная, глобальная интерполяция; сплайны – Катмулла-Рома, линейный, кубический; полиномами – Лагранжа, Ньютона, Эрмита и Гаусса;
- ускорение обработки первичной информации;
- повышение точности расчетов и достоверности обработки данных;
- повышение степени аналитичности получаемых данных;

- сокращение вероятности возникновения расхождения фактических и формальных показателей;
- повышение точности расчетов результатных показателей.

Системные требования к инструментальному средству.

Программа выполняется на IBM PC-совместимых компьютерах под управлением операционной системы Microsoft Windows 7/8/10. Работа Программы заключается в постоянном взаимодействии с пользователем (студентом) посредством диалоговых форм.

Программа должна работать без прерываний при возникновении критических ошибок, даже тех, которые возникают по вине пользователя или из-за ошибок во входных данных. Соответственно, секции программы, в которых возможно возникновение таких ошибок должны обрабатываться в программе особым образом. В случае возникновения ошибки после выдачи соответствующего предупреждения программа должна продолжить свою работу.

Программное обеспечение, реализующее работу «Интерполяция функций» может быть представлено в виде destop-приложения.

Программное обеспечение, реализующее работу «Интерполяция функций» должно обеспечивать выполнение следующий внутренних функций:

- программа должна обладать удобным интерфейсом;
- оператору должна быть предоставлена возможность руководить ходом ввода информации и получение графиков и результатов интерполяции;
- тестирование и испытание программного обеспечения должны проводиться в соответствии с указанными требованиями;
- разрабатываемое программное обеспечение должно быть надежным и устойчиво выполнять свои функции;

- разработчик должен совершать сопровождение разработанной программной системы.

Требования по производительности.

К разрабатываемому программному обеспечению сформулированы следующее требования по производительности: время реакции системы на запрос пользователя не более 1с.

Требования по интерфейсу.

Разрабатываемая программа должна обеспечить устойчивое функционирование.

Модуль программной системы должен быть выполняемым и представленным как *.exe программа.

Операционные требования.

При работе «Интерполяция функций» должна быть предусмотрена возможность управления работой системы при помощи запуска оператором программных процедур в реальном масштабе времени, а также прерывать их работу.

Требования по переносимости.

Разрабатываемое программное обеспечение должно удовлетворять требованиям мобильности, то есть способности работать в операционных системах Microsoft Windows 7/8/10, на различных компьютерах и носителях без изменения самого программного обеспечения.

Требования по качеству.

Разрабатываемое программное обеспечение должно быть корректным, эффективным, гибким, защищенным, достоверным и простым в применении.

При возникновении ошибки ввода вывода программа должна выводить сообщение на экран компьютера.

Автономные требования к инструментальному средству.

При эксплуатации в среде, соответствующей требованиям настоящего технического задания, подготовка программы к работе (загрузка) не может превышать 5 секунд с момента её запуска, а ответная реакция на действия

пользователя в процессе работы должна быть сформирована не позднее чем через 0,5 секунды.

Программа может допускать использование динамически подключаемых библиотек (DLL), не включённых в состав операционной системы. В этом случае динамические библиотеки включаются в состав поставки программы. При эксплуатации программы динамические библиотеки должны быть размещены в директории исполняемого модуля или зарегистрированы в операционной системе соответствующим образом.

Требования к составу и параметрам технических средств.

Программа выполняется под управлением операционной системы MS Windows на IBM PC-совместимом компьютере, соответствующем следующим характеристикам:

1. рекомендуется процессор Intel Pentium/Celeron, AMD K6/Athlon/Duron или совместимым с частотой 300 МГц или более (одно- или двухпроцессорная система). Минимальная частота процессора – 233 МГц;
2. рекомендуется 128 МБ ОЗУ или более. Минимально допустимый объем – 64 МБ (при наличии 64 МБ ОЗУ возможно снижение производительности и функциональности);
3. 200 МБ свободного места на жестком диске;
4. монитор и видеоадаптер Super VGA с разрешением 800 x 600 или более высоким;
5. дисковод для компакт- или DVD-дисков;
6. клавиатура и мышь Microsoft или совместимое указывающее устройство.

Требования к обеспечению надежного функционирования к инструментального средства.

Надежное (устойчивое) функционирование Программы должно быть обеспечено выполнением следующих организационно-технических мероприятий:

- организацией бесперебойного питания технических средств;
- использованием лицензионного программного обеспечения.

Отказы из-за некорректных действий пользователей системы.

Отказы Программы вследствие некорректных действий пользователя при взаимодействии с программой недопустимы. В случае некорректных действий пользователя программа выводит сообщение об ошибке и прекращает работу до её устранения пользователем.

Отказы Программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА ИНТЕРПОЛИРОВАНИЯ ФУНКЦИЙ

2.1 Обзор и анализ инструментальных средств поддержки

Для реализации функций системы необходима программная реализация функций автоматизированной системы. Рынок программных средств в настоящее время многообразен, как в области готовых прикладных решений, так и в области средств разработки. Однако, поиск готовых программных решений в рассматриваемой предметной области желаемых результатов не дал. Поэтому для реализации функций программной системы необходима разработка нового программного продукта, который способен гибко реализовать необходимые действия автоматизированной системы.

Для обеспечения разработки возникает задача выбора оптимальных программных и информационных средств для реализации функций системы.

При создании программного продукта основными критериями выбора средств разработки являлись:

- удобство использования;
- скорость разработки приложений и работы программы.

Как дополнение к перечисленному, можно указать, что время разработки зависит от: предоставляемого программного инструмента, наличия предварительного опыта у разработчиков в использования соответствующих программных средств и наличия необходимых для их эффективного использования аппаратных средств. Обеспечить минимальное время разработки можно только при выполнении этих условий.

Исходя из приведенных требований, выделим следующие характеристики средств разработки программного обеспечения и оценим их важность для рассматриваемого проекта по пятибалльной шкале.

1. Наличие опыта разработки с использованием данного программного продукта. (Цель – максимизировать показатель, весовой коэффициент “5”). Оценка “1” соответствует отсутствию опыта у разработчика системы, оценка 5 соответствует большому опыту при использовании данного продукта, Оценка 3 характеризует средний опыт разработчика.

2. Требования к вычислительным ресурсам (Цель – максимизировать показатель, весовой коэффициент 5) Оценка “1” соответствует “высоким требованиям”, оценка “3” средним требованиям, оценка “5” - низким требованиям к ресурсам машины.

3. Предоставляемые возможности создания интерфейса. (Цель – максимизировать показатель, весовой коэффициент 5). Оценка “1” соответствует отсутствию возможности проектировать современный интерфейс, оценка “5” соответствует широкому спектру предоставляемых возможностей. Оценка “3” характеризует средний уровень возможностей по проектированию интерфейса.

4. Скорость работы разработанного программного обеспечения. (Цель – максимизировать показатель, весовой коэффициент 5). Оценка “1” соответствует низкой скорости, оценка 5 соответствует высокой скорости. Оценка “3” характеризует средний уровень скорости.

По требованиям заказчика разрабатываемая автоматизированная система должна работать под операционной системой Windows. Поэтому ограничим выбор программных средств, удовлетворяющих этому требованию. Среди современных доступных средств разработки для операционной системы Windows можно выделить наиболее часто используемые:

– Delphi Embarcadero;

– C#-SQLite;

– QT;

– Microsoft Visual Studio Express.

Каждое из этих средств содержит весь спектр современных средств разработки. Главное отличие состоит в области использования рассматриваемых средств. Так Delphi Embarcadero обычно используется при разработке приложений, выполняющих большое количество вычислений. Основными недостатками этого программного продукта являются высокие требования к системным ресурсам и медленная скорость компиляции исходного кода. Система разработки QT предъявляет наименьшие требования к системным ресурсам. Основное достоинство Delphi Embarcadero состоит в предоставлении разработчику большого количества визуальных компонентов для разработки интерфейса и высокая надежность системы. Для сравнения этих программных продуктов воспользуемся методом вариантных обоснований. Этот метод предназначен для выбора наилучшего варианта из нескольких предложенных и состоит из следующих этапов:

– определение критериев, по которым будет произведено сравнение и степени их важности;

– каждый вариант оценивается по полученному перечню критериев. В результате получается значение – количественная оценка показателя;

– нахождение общей суммы баллов для каждого из вариантов (можно учитывать важность критериев);

– лучшим считается вариант, который набрал максимальное количество баллов.

Результаты приведены в таблице 2.1.

Таблица 2.1

Результаты выбора средств разработки

Средство разработки	**Характеристика**				**Сумма**
	1 (5)	2 (5)	4 (5)	5 (5)	
Delphi Embarcadero	5	4	4	5	100
C#-SQLite	2	3	5	4	80
QT	2	2	4	2	65
Microsoft Visual Studio Express	2	4	3	4	90

В результате проведенного исследования получили, что лучшим инструментальным средством для разработки системы Delphi Embarcadero.

Наряду с традиционными инструментами доступа к данным можно применять технологию Microsoft ActiveX Data Objects (ADO), которая основана на возможностях COM, а именно интерфейсов OLE DB.

2.2 Реализация алгоритмов интерполяции

Линейная интерполяция

При линейной интерполяции точки x_i располагаются в порядке возрастания и на каждом интервале от x_{i-1} до x_i i=1,2,…,n кривая $f(x)$ заменяется отрезком прямой линии, соединяющей точки x_{i-1}, y_{i-1} и x_i, y_i. Коэффициенты *a* и *b* уравнения прямой находятся из соотношения:

$$\frac{y-y_{i-1}}{x-x_{i-1}} = \frac{y_i-y_{i-1}}{x-x_{i-1}}, \text{ откуда } y = \frac{y_i-y_{i-1}}{x-x_{i-1}} * x + y_{i-1} - \frac{y_i-y_{i-1}}{x-x_{i-1}} * x_{i-1}. \quad (2.1)$$

И коэффициенты:

$$a = \frac{y_i - y_{i-1}}{x - x_{i-1}}, b = y_{i-1} - a * x_{i-1}$$

Если значение x не попадает в интервал$[x_0, x_n]$, значение функции y определяется путем экстраполяции. Если $x < x_0$, то y определяется по уравнению прямой на участке $[x_0, x_1]$. Если же $x > x_n$ – по уравнению прямой на участке $[x_{n-1}, x_n]$ (рис. 2.1).

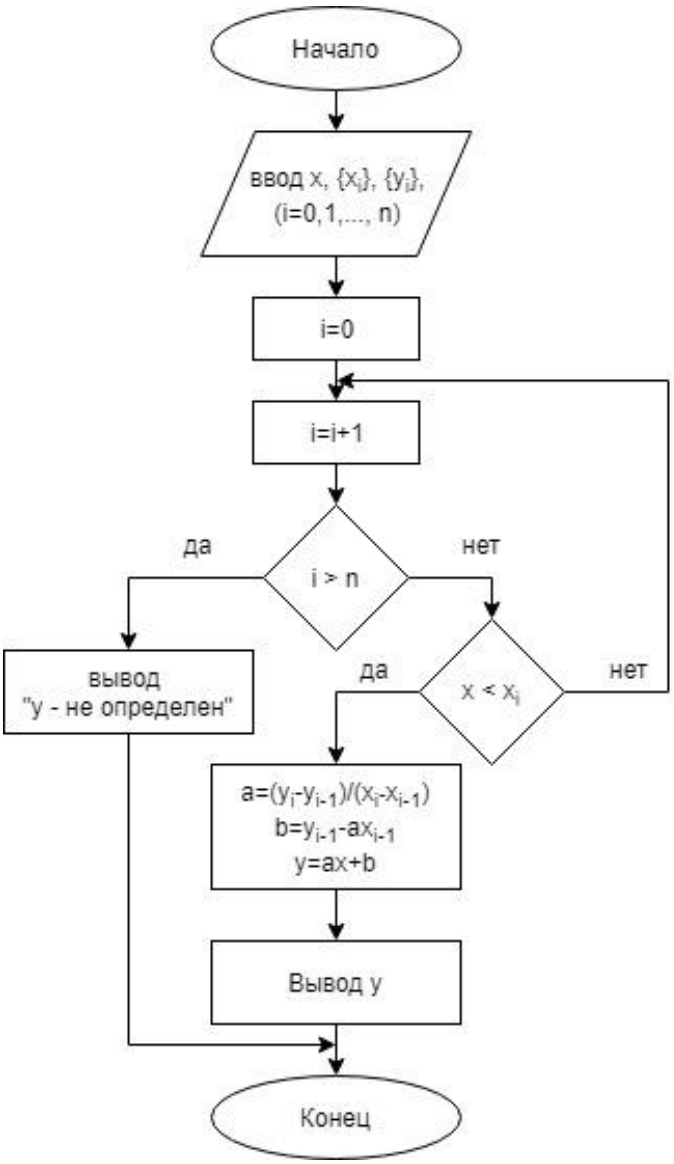

Рис. 2.1. Блок-схема линейной интерполяции

Квадратичная интерполяция

Вычислить коэффициенты C_0^k, C_1^k, C_2^k параболы $R_k = C_2^k x^2 + C_1^k x + C_0^k$, проходящей через три узла интерполяции (x_1,$R(x_1)$), (x_2, $R(x_2)$) и (x_3, $R(x_3)$), решив систему: $\begin{cases} R_k(x_1) = R(x_1) \\ R_k(x_2) = R(x_2) \\ R_k(x_3) = R(x_3) \end{cases}$

Вычисляем аналитически положение экстремума: $x_k^* = -\frac{C_1}{2C_2}$.

Проверяем выполнение условия $| x_2 = x_k^* | \leq \varepsilon$. Если условие не выполняется, то задаём $x_2 = x_k^*$. В противном случае считаем x^*_k найденным с заданной погрешностью ε точкой экстремума, вычисляем $R(x_k^*)$ и останавливаем счёт. Если условие не выполняется, то задаём $x_3 = x_2$, если $x_1 < x_k^* < x_2$, или $x_1 = x_2$, если $x_2 < x_k^* < x_3$, $x_2 = x_k^*$. Блок-схема метода показана на рисунке 2.2

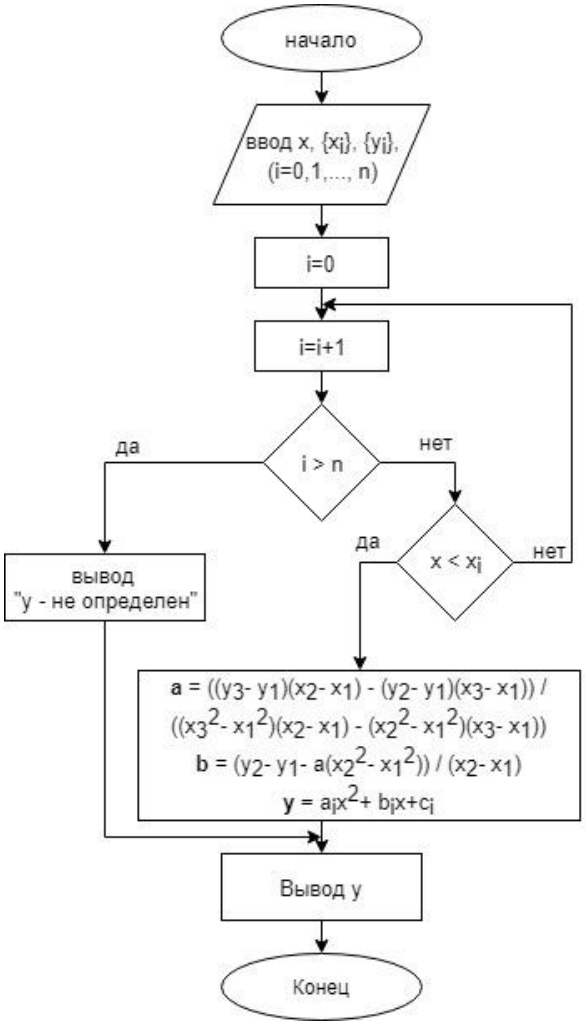

Рис. 2.2. Блок-схема квадратичной интерполяции

Кубический сплайн

На каждом отрезке $[x_{i-1}, x_i]$ функция *S(x)* есть полином третьей степени $S_i(x)$, коэффициенты которого надо определить. Дополнительно требуется непрерывность первой и второй производных функции S(x) на отрезке $[x_0, x_n]$. По формуле:

$$S_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3, \qquad (2.4)$$

отсюда:

$$S_i(x) = a_i,\ S_i'(x_i) = b_i,\ S_i''(x_i) = c_i. \qquad (2.5)$$

Если u(x) непрерывна, то последовательность кубических сплайнов $U_N(x)$ будет сходиться к u(x) равномерно (рис. 2.3).

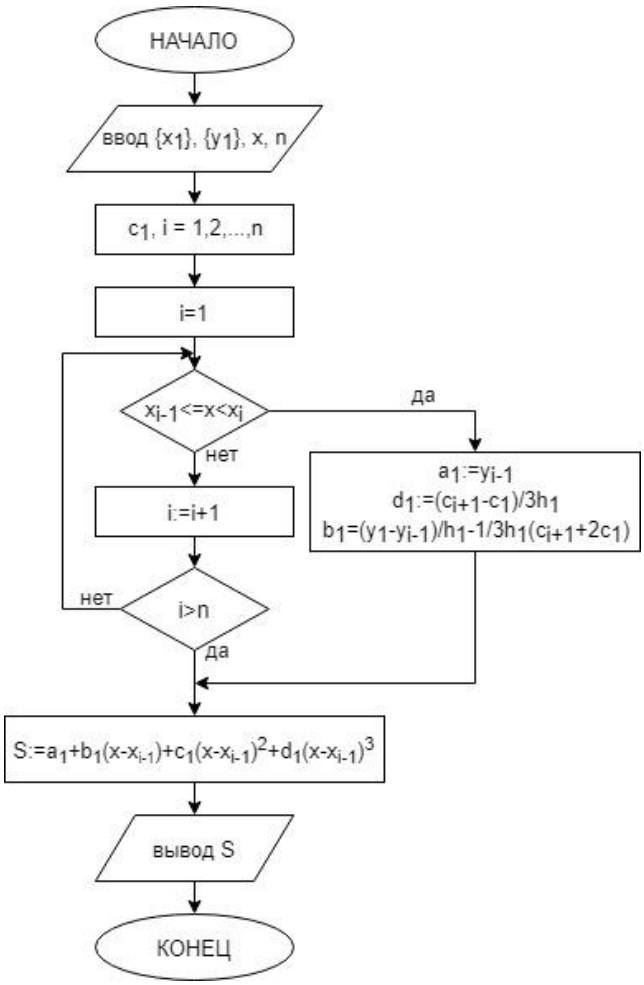

Рис. 2.3. Блок-схема интерполяции кубическим сплайном

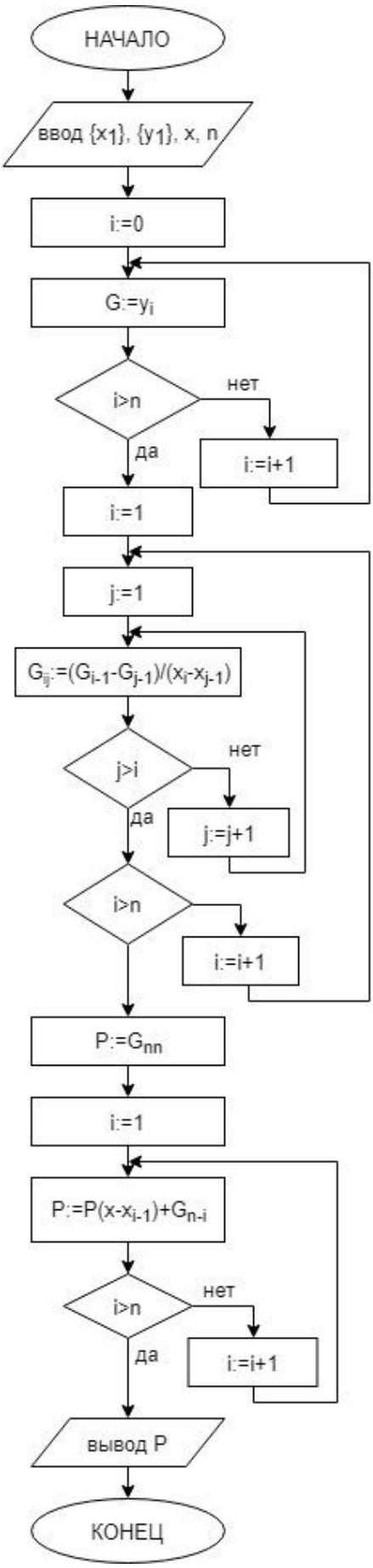

Рис. 2.4. Блок-схема полинома Ньютона

Полином Ньютона

Формула полинома:

$$L_n(x) = \sum_{i=0}^{n} b_1(x - x_i)(x - x_2) \dots (x - x_i), \quad (2.6)$$

где

$$b_0 = f(x_i), b_1 = [x_1, x_2]f, b_2 = [x_1, x_2, x_3]f, \dots, b_n = [x_1, x_2, \dots, x_{n+1}]f \quad (2.7)$$

где

$$d_{n+1} = f(x_{n+1}), d_n = [x_n, x_{n+1}]f, d_{n-1} = [x_{n-1}, x_n, x_{n+1}]f, d_1 = [x_1, x_2, \dots, x_{n+1}] \quad (2.8)$$

Блок-схема изображена на рисунке 2.4

Полином Лагранжа

При глобальной интерполяции на всем интервале $[a,b]$ строится единый многочлен. Одной из форм записи интерполяционного многочлена для глобальной интерполяции является многочлен Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} y_i * l_i(x) \quad (2.9)$$

где $l_i(x)$ – базисные многочлены степени *n*:

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} \quad (2.10)$$

То есть многочлен Лагранжа можно записать в виде:

$$L_n(x) = \sum_{i=0}^{n} y_i * \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \quad (2.11)$$

Многочлен $l_i(x)$ удовлетворяет условию $l_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$. Это условие означает, что многочлен равен нулю при каждом x_j кроме x_i, то

есть $x_0, x_1 \dots x_{i-1}, x_{i+1}, \dots x_n$ – корни этого многочлена. Таким образом, степень многочлена $L_n(x)$ равна n и при $x \neq x_i$ обращаются в ноль все слагаемые суммы, кроме слагаемого с номером $i = j$, равного y_i (рис. 2.5).

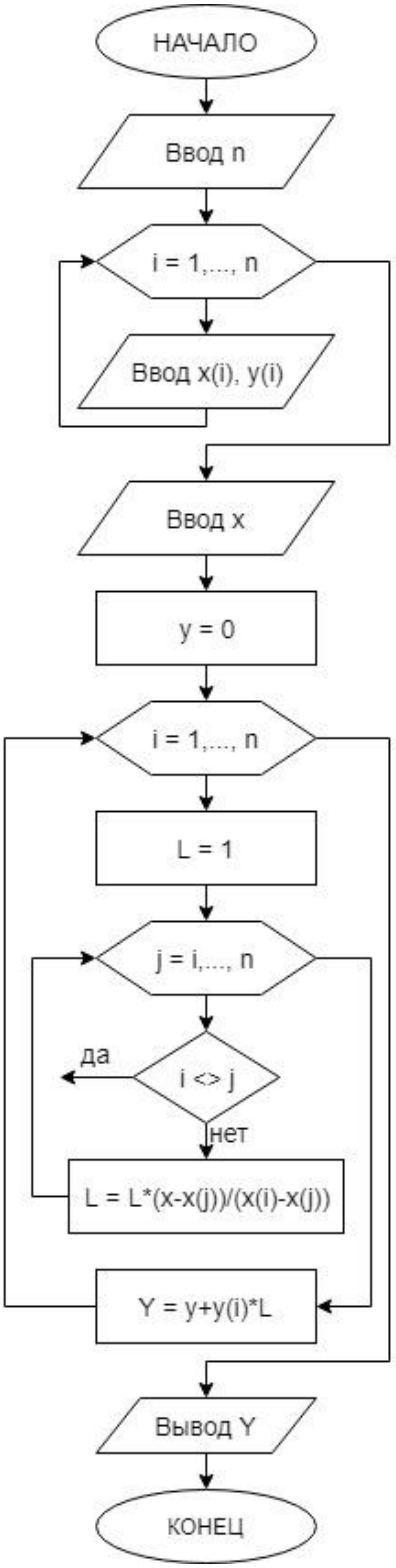

Рис. 2.5. Блок-схема полинома Лагранжа

Полином Гаусса

Формула, использующая в качестве узлов интерполяции ближайшие к точке интерполирования x узлы (рис. 2.6). Если $x = x_0 + th$, то формула имеет вид:

$$G_{2n+1}(x_0 + th) == f_0 + f_{\frac{1}{2}}^{1} t + f_0^2 \frac{t(t-1)}{2!} + \cdots + f_0^{2n} \frac{t(t^2-1)\ldots[t^2-(n-1)^2](t+n)}{(2n)!} \quad (2.12)$$

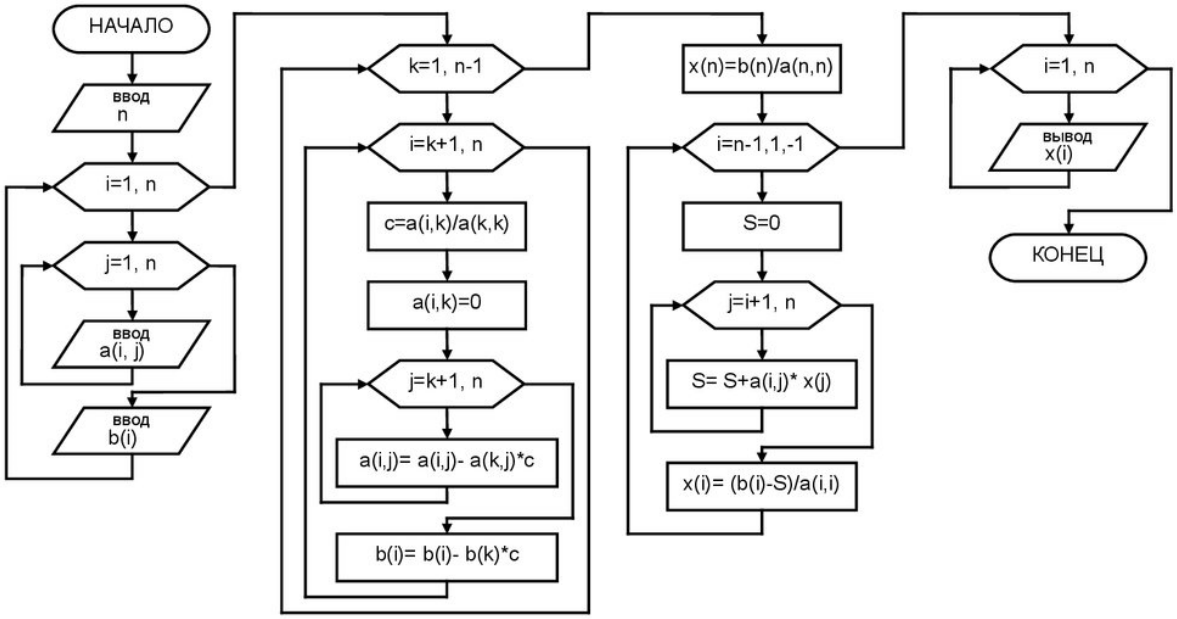

Рис. 2.6. Блок-схема интерполяции методом Гаусса

Полином Эрмита

Полином, служащий для вычисления интерполируемой функции в соответствующем интервале, имеет вид:

$$p(t) = (2t^3 - 3t^2 + 1)p_k + (t^3 - 2t^2 + t)(x_{k+1} - x_k)m_k + \\ +(-2t^3 - 3t^2)p_{k+1} + (t^3 - t^2)(x_{k+1} - x_k)m_{k+1} \quad (2.13)$$

Для вычисления значений первой производной используются различные способы. Простейшим является вычисление среднего

арифметического значения разделенных первых разностей на двух соседних интервалах.

$$m_k = \frac{p_{k+1} - p_k}{2(t_{k+1} - t_k)} + \frac{p_k - p_{k-1}}{2(t_k - t_{k-1})} \qquad (2.14)$$

В так называемом кардинальном сплайне используется формула:

$$m_k = (1 - c)\frac{p_{k+1} - p_{k-1}}{t_{k+1} - t_{k-1}} \qquad (2.15)$$

В этой формуле параметр *c* изменяется от 0 до 1. В соответствии с этой формулой производная в середине отрезка равняется разделенной первой разности на всем отрезке, умноженной на некий коэффициент (рис.2.7).

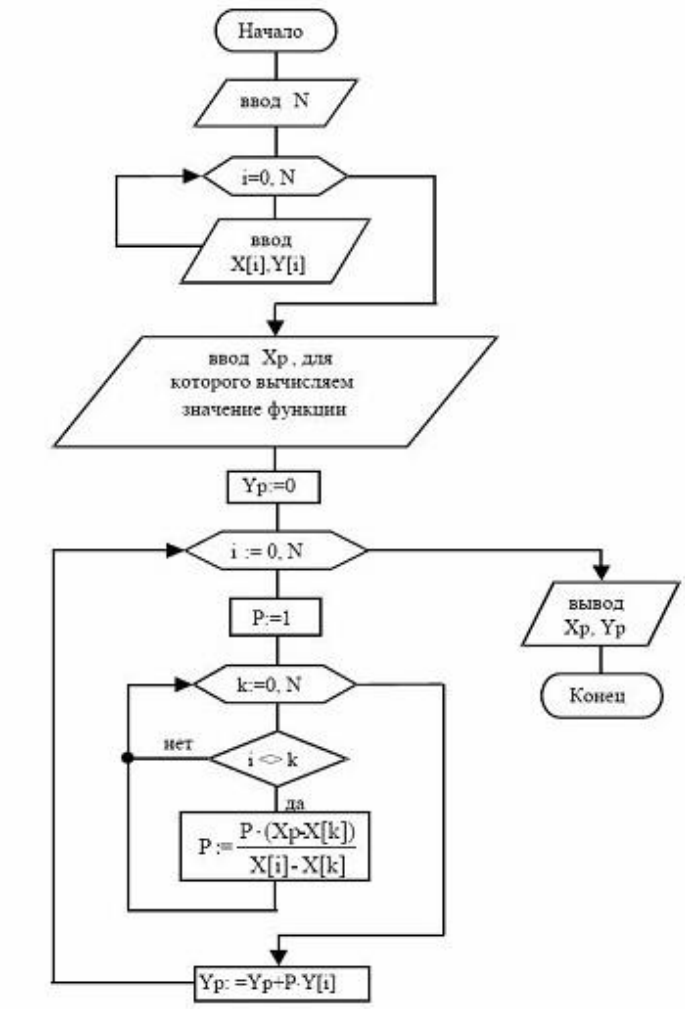

Рис. 2.7. Блок-схема полинома Эрмита

2.3 Модульная структура инструментального средства интерполирования функций

Программное обеспечение состоит из 4 модулей, схема приведена на рисунке 2.8.

Модуль BuildForm_Int отвечает за сбор введенной информации воедино, проверку ошибок ввода текущих параметров интерполяции. Сбор данных осуществляется процедурой нажатия кнопки «Построить»: procedure TprjDataInput.btnAcceptClick. Данная процедура преобразует и присваивает значения из полей ввода данных. Если значение XValue или массив ArrValuesF пусты, происходит вызов функций генерации значений: function GenerateXValue (XValue:real):real и function GenerateArray ():String.

После сбора данных, происходит вызов процедур модуля главной формы:

MainForm_Int.InitializeBuildingGraph и MainForm_Int.AnalyzeFunction для просчета значений и анализа формулы.

Модуль MainForm_Int отвечает за отображение введенных параметров интерполяции, очистку графиков и вывод просчета значений x. Основная процедура модуля: procedure InitializeBuildingGraph. В ней производится просчет первоначальных значений для f(x) и таблицы, а также, - перебор массива видов интерполяции, после чего, в свою очередь, происходит обращение к модулю Methods_Int для построения графика интерполяции.

Модуль ValueForm_Int отвечает за повторный ввод/изменение значений массива/точки x. В него входит процедура присвоения значений.

Модуль Methods_Int отвечает, непосредственно за рисование графиков, просчет параметров интерполяции. И имеет следующие процедуры:

- procedure LinearI - Линейная интерполяция;
- procedure Quadr - Квадратичная интерполяция;
- procedure Global - Глобальная интерполяция;

- procedure LinearSpline - интерполяция линейным сплайном;
- procedure QuadrSpline - интерполяция кубическим сплайном;
- procedure CatmullRomSpline - интерполяция сплайном Катмулла-Рома;
- procedure PolGauss - интерполяция полиномом Гаусса;
- procedure PolErmita - интерполяция полиномом Эрмита;
- procedure Lagranj - интерполяция полиномом Лагранжа;
- procedure Newton - интерполяция полиномом Ньютона;

Все процедуры получают следующие параметры:

- XValue:real - Введенное значение X;
- ArrValuesF:TDoubleDynArray; - Массив значений X;
- ArrYValuesF:TDoubleDynArray; - Массив значений Y;
- Calculate: Boolean - выводить ли просчет в таблицу;

Рис. 2.8. Модульная схема приложения

2.4 Разработка инструментального средства интерполирования функций

2.4.1 Разработка интерфейса программы

На рисунке 2.10 изображено главное окно разработанного приложения. При первом запуске оно заполняется тестовыми значениями. Снизу окна расположено две таблицы TStringGrid: справа - для вывода расчетов интерполяции и погрешности; слева - для отображения введенных пользователем значений массива «x» и рассчитанных значений «y», соответственно. Выше находится блок с введенными параметрами расчета. Справа вверху находится компонент «График» TChart и его легенда. Слева вверху - командная панель.

В таблице 2.2 приведены кнопки TBitBtn, доступные на командной панели, и их функции.

Таблица 2.2

Функции кнопок на командной панели

Кнопка	Значение
Провести интерполяцию	Выводит вторую основную форму приложения, на которой можно ввести основные параметры интерполяции, а также, - выбрать ее виды.
Вычислить значения функции	Выводит на экран форму ввода значения «x».
Изменить значения таблицы	Выводит на экран форму ввода значений массива.
Очистить график	Очищает график функции.
Закрыть приложение	Закрывает приложение «Интерполяция функции».

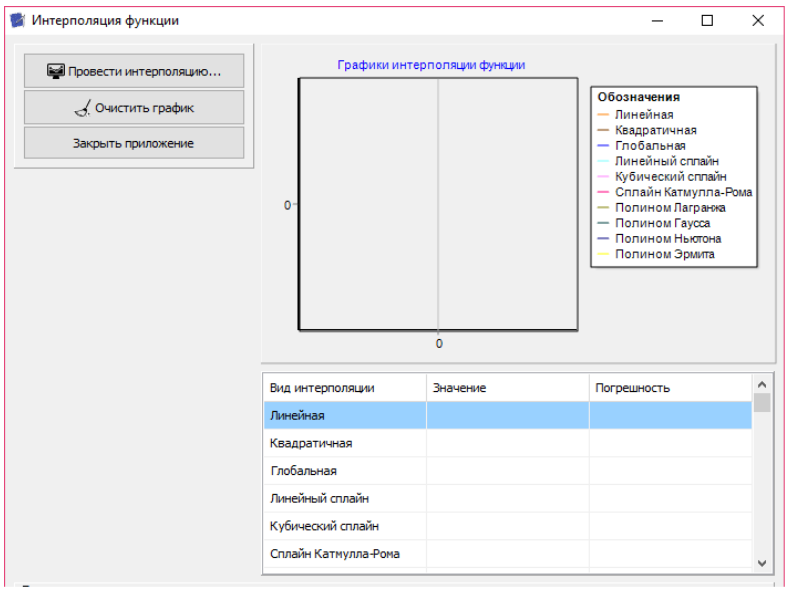

Рис. 2.10. Главное окно приложения

На рисунке 2.11 изображено окно ввода первоначальных данных для интерполяции. Слева находится компонент TCheckListBox с видами доступных интерполяций. TCheckBox ниже, - позволяет единовременно поставить/снять пометку каждому элементу TCheckListBox. В правой части окна расположен TComboBox с выбором функции (на данный момент доступна одна функция, но в планах доработать функционал до возможности ввода функции вручную с последующим сохранением списка в конфигурационном файле).

Ниже расположен компонент TEdit, предназначенный для ввода значений одномерного массива (обязательные условия – 6 значений, разделенных запятой, без пробелов). На данном этапе реализовано несколько проверок ввода значений: удаляются все знаки после последнего числа; удаляются точки, которые расположены до/после запятых; удаляется вторая точка в дробном числе, если таковая имеется; проверка ввода двух нулей подряд, если впереди запятая; исключение ошибки пустого поля. Проверки осуществляются при потере компонентом фокуса. Далее находится элемент TMemo, который несет сугубо информативный характер. После него расположены два компонента TCheckBox: один очищает график, другой –

производит вычисления функции по заданному «X». Ниже находится TEdit, позволяющий присвоить значение переменной «X» и компоненты TBitBtn, отвечающие за Запись в файл и чтение из файла соответственно. Когда все параметры будут заполнены, следует нажать на компонент TBitBtn «Построить» или отказаться от сделанных настроек, нажав на TBitBtn «Отмена».

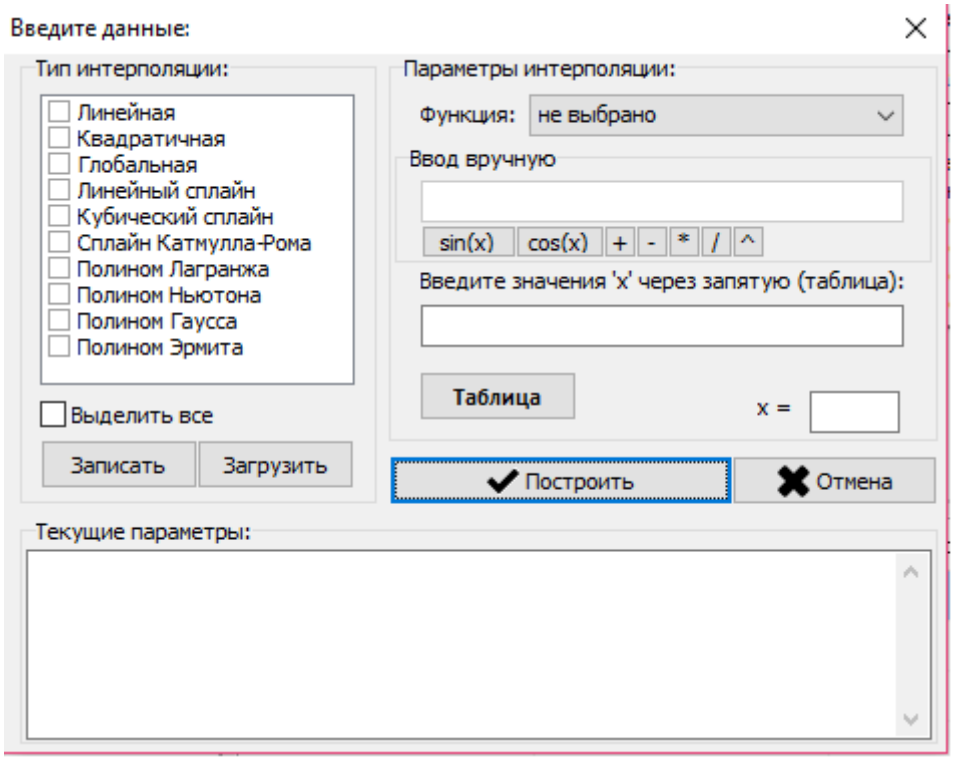

Рис. 2.11. Окно ввода данных

Когда пользователь приложения нажмет на TBitBtn «Построить», приложение соберет всю конфигурационную информацию и выведет в информационном окне, которое изображено на рисунке 2.12.

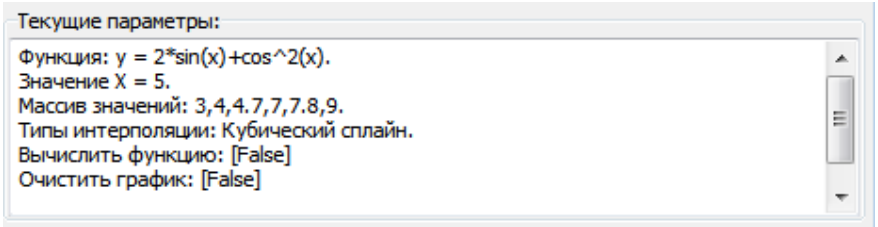

Рис. 2.12. Информационное окно с выбранной конфигурацией

Если пользователь не выберет ни единого вида интерполяции, приложение это сделает за него, выбрав один вариант из списка, но прежде, - выдаст сообщение, которое изображено на рисунке 2.13.

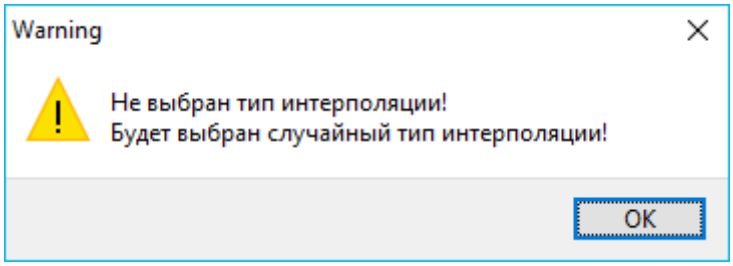

Рис. 2.13. Ошибка выбора интерполяции

2.4.2 Разработка алгоритма головной программы

Схема алгоритма головной программы описывает общий сценарий работы разрабатываемого приложения (прил.2). В составе проекта приложения предусматривается три формы:

1) главная форма приложения;

2) форма ввода значений X и функции f(x);

3) форма ввода значений X и Y.

При запуске приложения отображается главная форма, на которой находятся управляющие элементы: поля для вывода интерполируемых значений и панель управления приложением. В схеме головного алгоритма предусматривается обработка следующих событий:

1) Выбор формулы;

2) Выбор вида интерполяции;

3) Ввод массива значений;

4) Расчет;

5) Построение графика.

Проектирование алгоритма ввода и вывода данных.

При интерполяции табличной функции полиномом исходными данными являются:

- количество значений узловых точек n;
- таблица значений исходной функции X и Y;
- абсцисса искомой точки – xx.

Коэффициенты полинома хранятся в переменной-массиве.

3. ИСПЫТАНИЯ ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА

3.1 Программа и методика испытаний

Объектом испытаний, рассматриваемым в данной главе, является ранее спроектированная и разработанное «Инструментальное средство интерполяции функций».

Полный перечень требований к разрабатываемому приложению перечислен в третьем разделе первой главы, однако ключевыми пунктами, требующими установки соответствия, являются следующие:

- возможность интерполирования следующими методами: линейная, квадратичная, глобальная интерполяция; сплайны – Катмулла-Рома, линейный, кубический; полиномами – Лагранжа, Ньютона, Эрмита и Гаусса;
- работоспособность приложения на операционных системах Microsoft Windows 7/8/10;
- возможность ввода данных для интерполирования;
- возможность просмотра графиков по введенным данным;
- время отклика приложения не более 1 секунды;

Исходя из приведенных выше требований - был разработан следующий алгоритм для проведения испытаний автоматизированной системы.

1. Проверить работоспособность приложения.

   1.1 Запустить программу на операционной системе Windows 10.

   1.2 Протестировать модуль проведения интерполяции:

   - выбор метода интерполяции;
   - ввод размера массива точек;
   - ввод X и Y значений;
   - ввод X значений и функцию, которой задан Y;

- построение и отображения графика интерполяции по заданным данным;
- очистка графика.

1.3 Протестировать время отклика программы.

1.4 Проверка работоспособности программы при вводе ошибочных данных.

2. Запустить программу на операционной системе Windows 7 и выполнить все шаги из пунктов 1.2, 1.3 и 1.4.
3. Запустить программу на операционной системе Windows 8 и выполнить все шаги из пунктов 1.2, 1.3 и 1.4.

3.2 Проведение испытаний

Проведение испытаний приложения с описанным выше алгоритмом.

Согласно пункту 1.1 нужно проверить работоспособность программы в операционной системе Microsoft Windows 10, которая по своим параметрам и архитектуре не очень отличается от Microsoft Windows 8, на которой ранее проводилось тестирование. Если при запуске на Microsoft Windows 7 было заметно, как минимум, различие в графическом интерфейсе, то Microsoft Windows 8 даже визуально не отличается от Microsoft Windows 10.

Предыдущее тестовое испытание проводилось на Microsoft Windows 8 и показало, что приложение работает корректно, поэтому, следует ожидать, что на Microsoft Windows 10 тоже будет корректная работа программы.

На рисунке 3.1 изображен результат запуска приложения на операционной системе Microsoft Windows 10.

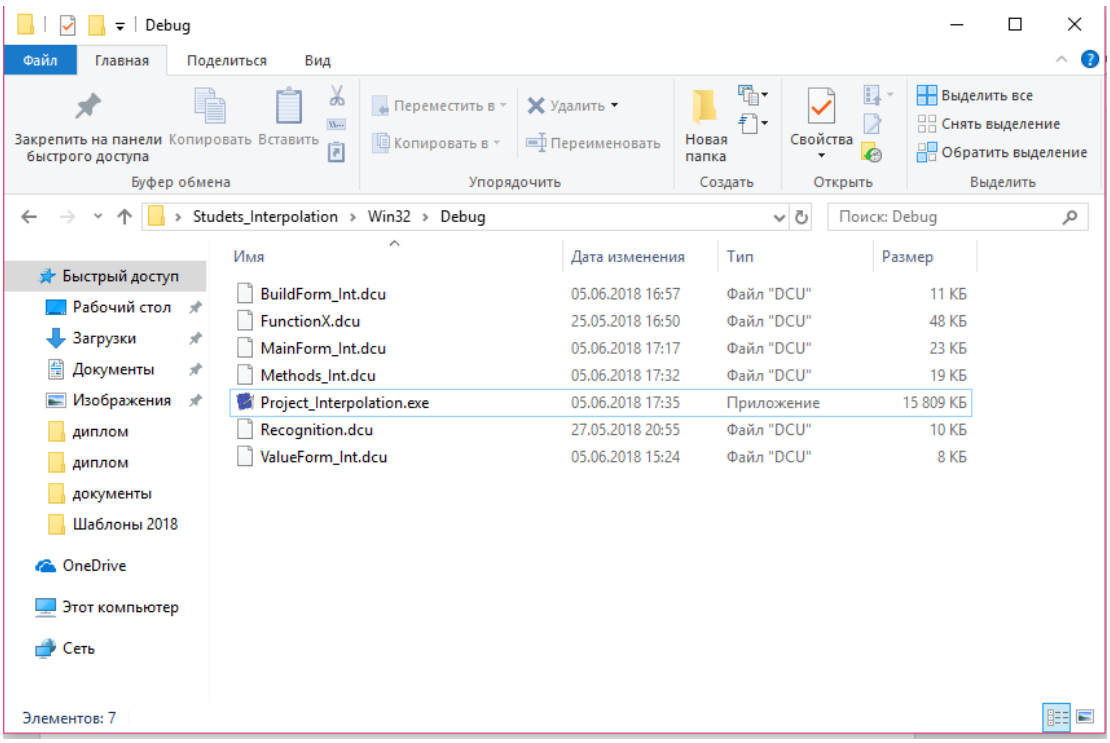

Рис. 3.1. Запуск приложения в MS Windows 10

После запуска приложения появляется головной модуль программы, который мы видим на рисунке 3.2.

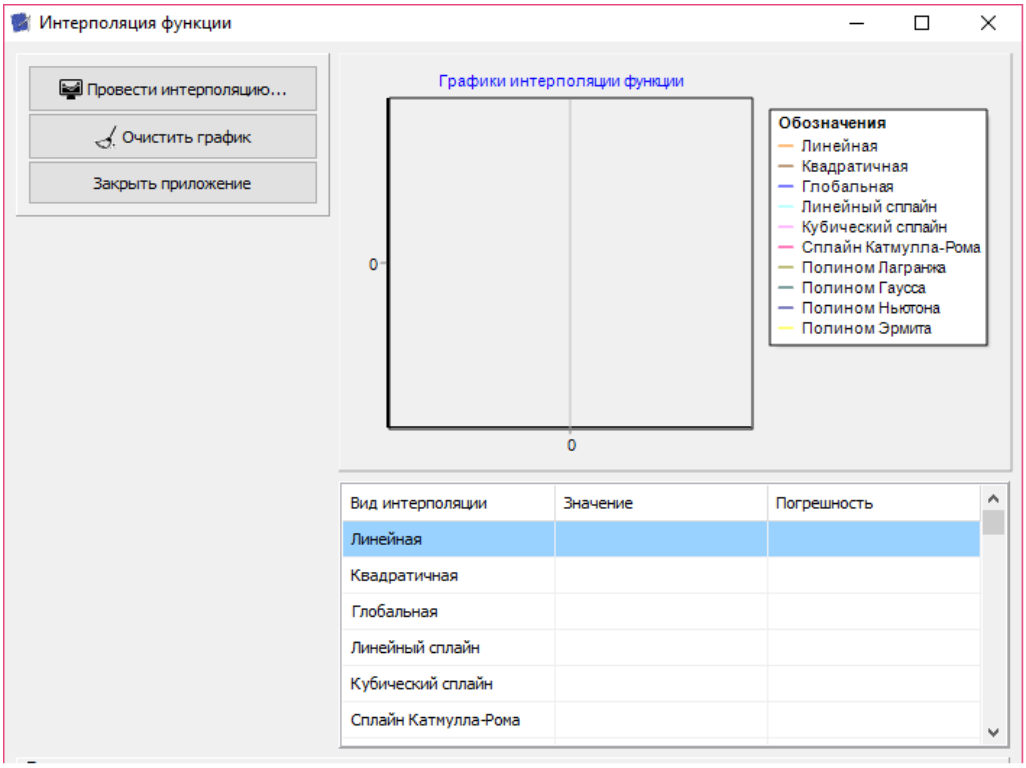

Рис. 3.2. Головной модуль приложения

Для проведения интерполирования нужно нажать на кнопку «провести интерполяцию», затем появится модуль для ввода данных, изображенный на рисунке 3.3.

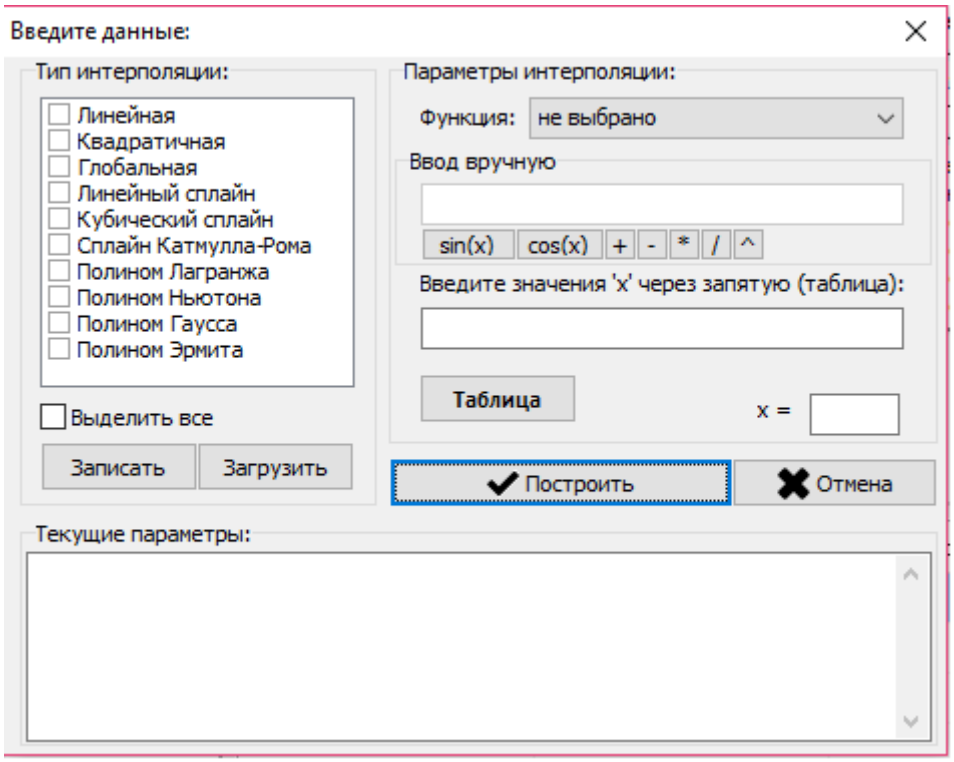

Рис. 3.3. Модуль ввода данных

Проведем тестирование приложения на примере линейной интерполяции, если интерполяция задана функцией f(x), то заполним окно ввода (рис. 3.4) следующими данными: X = 3,4,6,7,9,13

$$f(x) = 2*\cos(x)-(\sin(x))^2.$$

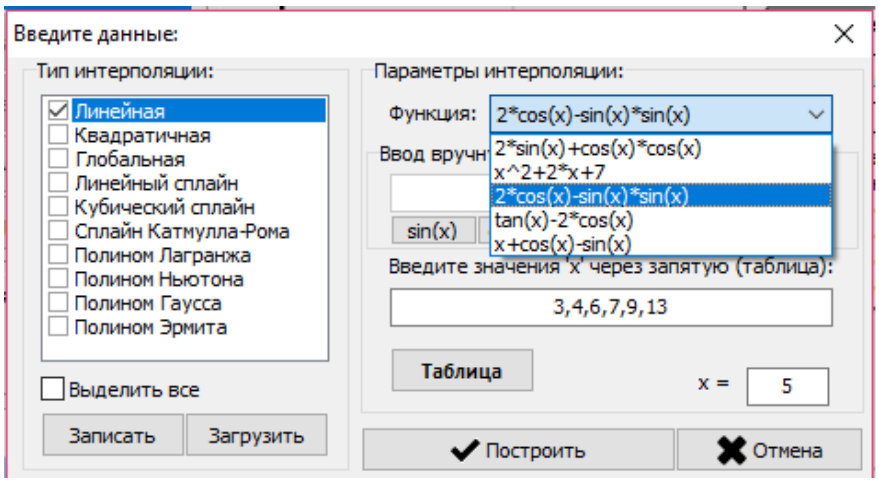

Рис. 3.4. Ввод значений X и функции f(x)

Здесь мы выбираем функцию из выпадающего списка, также можно выполнить ввод вручную, если функции не окажется в списке. Далее нажимаем кнопку «Построить». Если же функция задана таблично, то нажимаем кнопку «Таблица», появится окно для ввода табличных значений, как на рисунке 3.5.

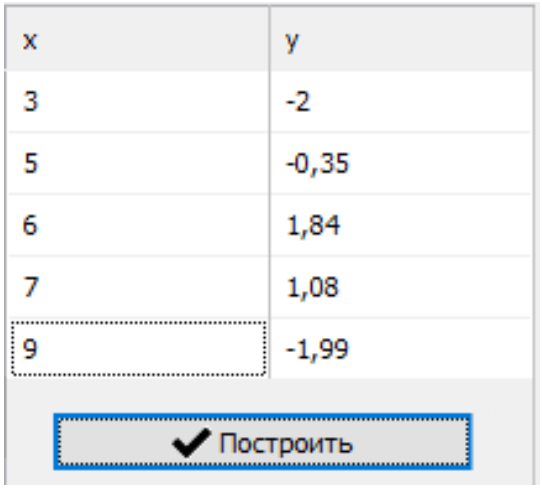

Рис. 3.5. Ввод значений таблично X и Y

Результат расчетов показан на рисунке 3.6.

Рис. 3.6 Линейная интерполяция

Для очистки графика нажимаем кнопку «Очистить график», результат показан на рисунке 3.7.

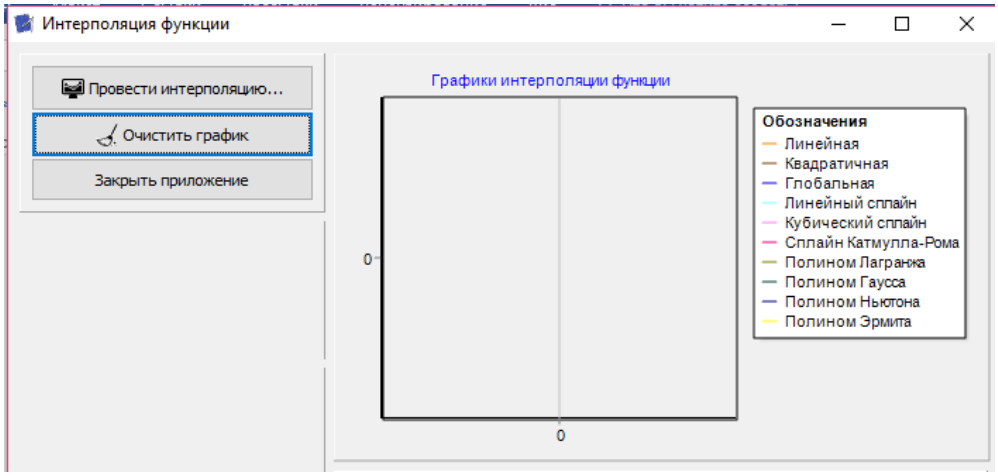

Рис. 3.7. Очистка графика

Испытание инструментально средства по пункту 1.3 показало, что приложение откликается мгновенно. Никаких зависаний и сбоев нет.

Испытание приложения при вводе ошибочных данных по пункту 1.4.

Если не выбран тип интерполяции, то программа выдает окно ошибки, как показано на рисунке 3.8.

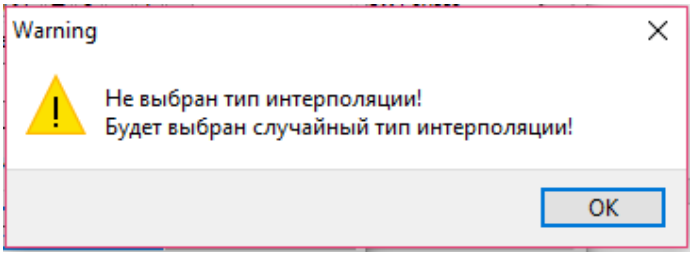

Рис. 3.8. Ошибка «Не выбран тип интерполяции»

Если же не будет введен массив «X», то программа выдаст следующее окно ошибки (рис.3.9):

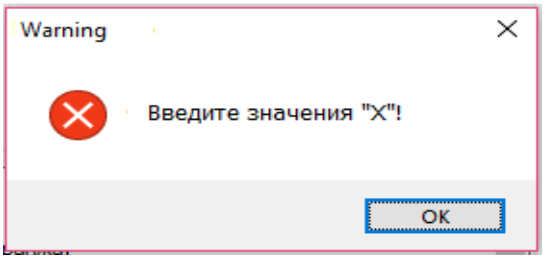

Рис. 3.9. Ошибка «Введите значения “X”»

Если ввести не все данные при заполнении таблицы значений X и Y, то программа выдаст ошибку, как на рисунке 3.10.

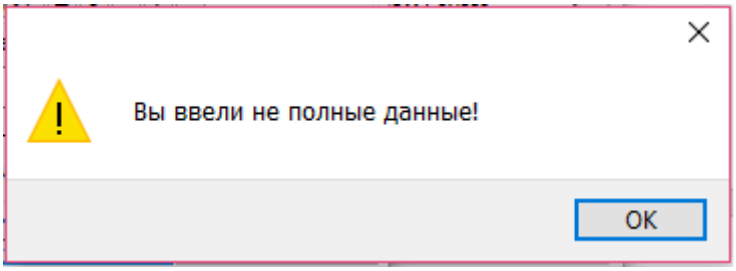

Рис. 3.10. Ошибка «Вы ввели не полные данные»

Согласно пункту 1.1 нужно проверить работоспособность программы в операционной системе Microsoft Windows 7. Окно запуска приложения приведено на рисунке 3.11.

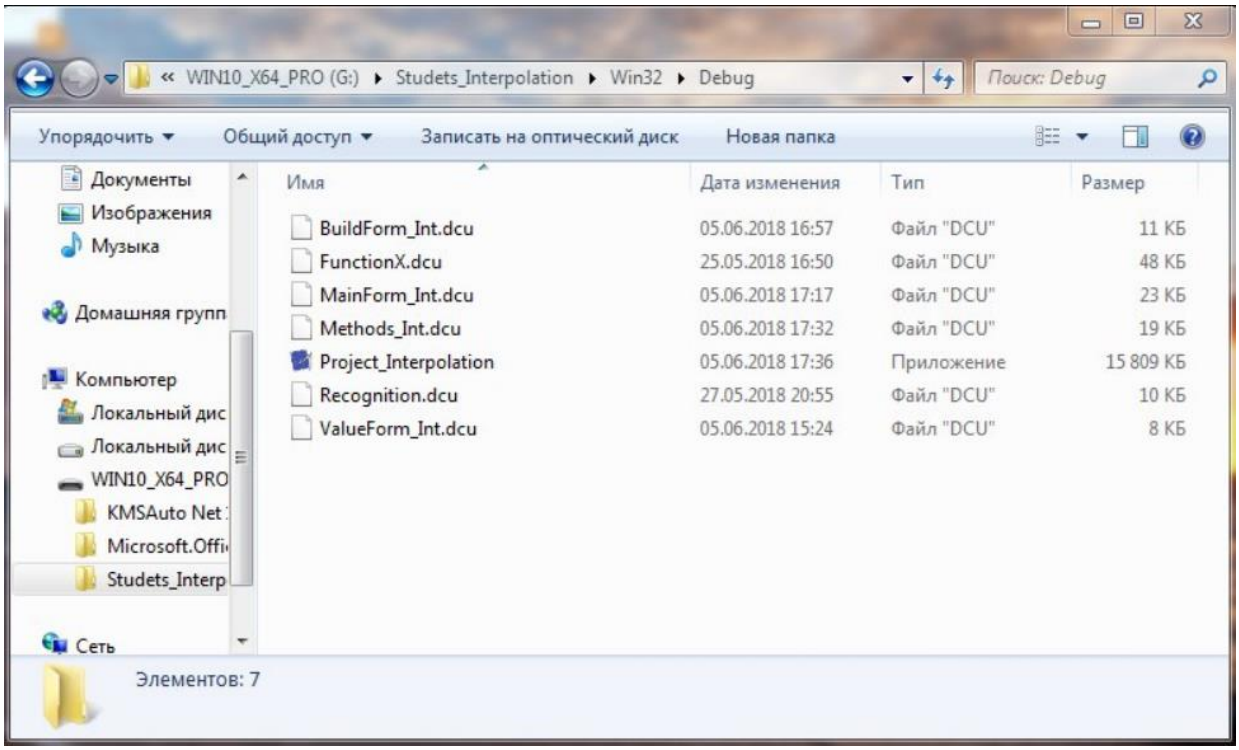

Рис. 3.11. Запуск приложения в MS Windows 7

После запуска приложения появляется головной модуль программы, который мы видим на рисунке 3.12.

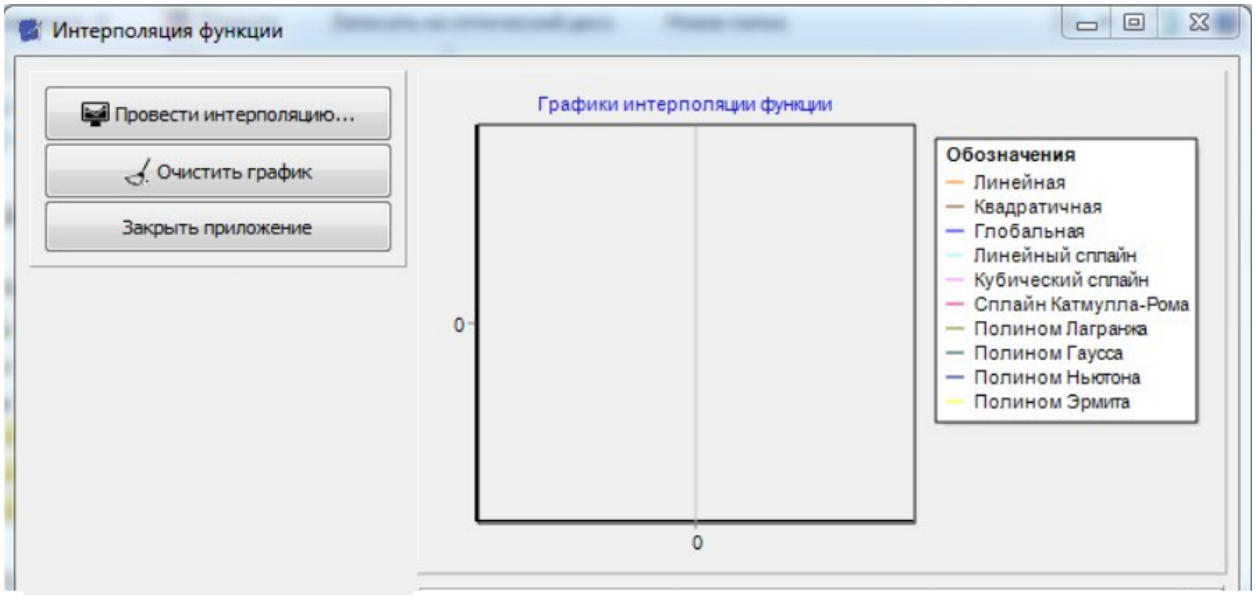

Рис. 3.12. Головной модуль приложения

Проведем тестирование приложения на примере линейной интерполяции, если интерполяция задана функцией f(x), то заполним окно ввода (рис. 3.4) следующими данными: X = 3,4,6,7,9,13

$$f(x) = 2*\cos(x)-(\sin(x))^2.$$

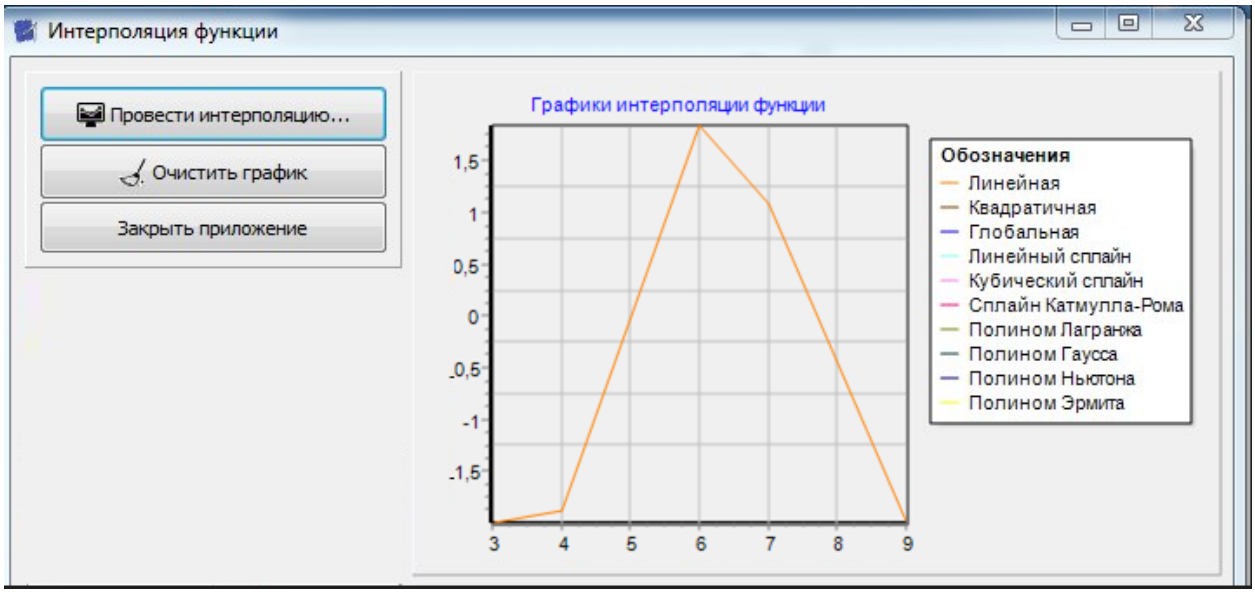

Рис. 3.13. Построение графика линейной интерполяции

3.3 Тестовые данные

Для каждого вида интерполяции существует минимальное количество точек, которое необходимо ввести. Данные приведены в таблице 3.1.

Таблица 3.1

Минимальное количество точек для каждого вида интерполяции

Тип интерполяции	Минимальное количество точек
Линейная	2
Квадратичная	8
Глобальная	10
Линейный сплайн	4
Кубический сплайн	8
Сплайн Катмулла-Рома	4
Полином Лагранжа	8
Полином Ньютона	4
Полином Гаусса	5
Полином Эрмита	4

На рисунке 3.14 показан график, на котором изображена линейная интерполяция функции по параметрам, приведенных в таблице 3.2.

Таблица 3.2

Параметры для линейной интерполяции

X	Y
3	1.26
4	-1.09
4.7	-2
7	-1.88
7.7	2
8.9	1.65

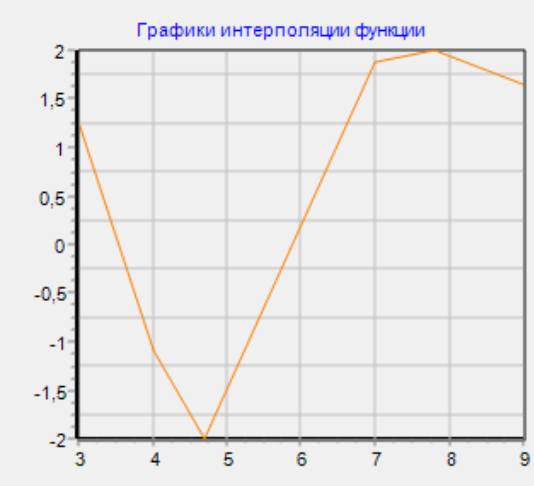

Рис. 3.14. Линейная интерполяция

На рисунке 3.15 показан график, на котором изображена квадратичная интерполяция функции при параметрах, приведенных в таблице 3.3. Если при линейной интерполяции каждые две точки соединяются прямыми, получая ломаную линию, то при квадратичной или, как ее еще называют, параболической, для построения используется уже три точки, из которых по параболической зависимости строится более точная интерполяция.

Таблица 3.3

Параметры квадратичной интерполяции

X	Y
3	1.26
4	-1.09
4.7	-2
7	-1.88
7.8	2
9	1.65
11	-2
13	1.66
15	1.88

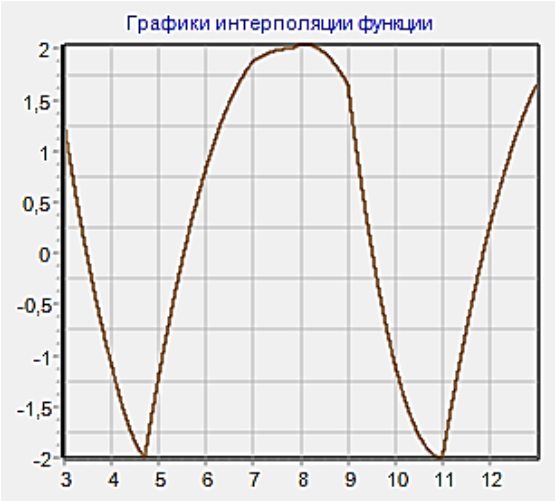

Рис. 3.15. Квадратичная интерполяция

На рисунке 3.16 показан график, на котором изображена глобальная интерполяция функции при параметрах из таблицы 3.4. Недостатком данной интерполяции является то, что с увеличением узлов интерполирования (n>10) может существенно ухудшится результат.

Таблица 3.4

Параметры глобальной интерполяции

X	Y
3	1.26
4	-1.09
4.7	-2
7	-1.88
7.8	2
9	1.65
11	-2
13	1.66
15	1.88
18	-1.07
32	1.8

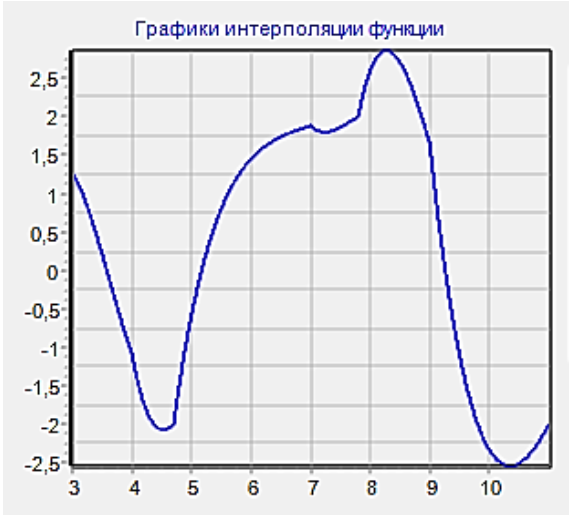

Рис. 3.16. Глобальная интерполяция

На рисунке 3.17 показан график, на котором изображена интерполяция функции линейным сплайном. В таблице 3.5 приведены параметры для интерполирования линейным и кубическим сплайном.

Таблица 3.5

Параметры для интерполяции линейным и кубическим сплайном

X	Y
3	1.26
4	-1.09
4.7	-2
7	-1.88
7.8	2
9	1.65
11	-2
13	1.66
15	1.88
18	-1.07
32	1.8
34	1.78
42	-1.67

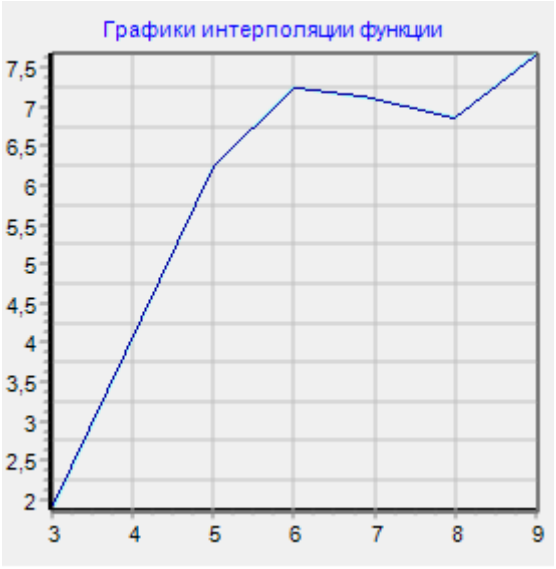

Рис. 3.17. Интерполяция линейным сплайном

На практике чаще требуется, чтобы интерполяция была задана кривой, а не ломаной линией. Для этого используется кубический сплайн, который строит график с помощью отрезков кубических парабол. В данном случае происходит аппроксимация между точками, которая влияет на построение кривой, потому что в промежутках между точками проводятся вычисления интерполирующей функции, чего не происходит при интерполировании линейным сплайном.

На рисунке 3.18 показан график, на котором изображена интерполяция функции кубическим сплайном.

Рис. 3.18. Интерполяция кубическим сплайном

На рисунке 3.19 показан график, на котором изображена интерполяция функции полиномом Лагранжа по параметрам, заданным в таблице 3.6. Данный вид интерполяции хорош тем, что проходит через каждую точку массива, но чем больше узлов в сетке, тем выше степень многочлена, поэтому требуется большее количество вычислений.

Таблица 3.6

Параметры для интерполяции полиномом Лагранжа

X	Y
3	1.84
4	2.47
4.7	-3.95
7	-0.64
7.8	18.4
9	1.37
15	0.66
27	-2.69
35	2.28
42	3.09
50	-2.2

Рис. 3.19. Интерполяция полиномом Лагранжа

На рисунке 3.20 показан график, на котором изображена интерполяция функции сплайном Катмулла-Рома по параметрам, заданным в таблице 3.7. Достоинством данного типа интерполирования является то, что кривая проходит через заданные контрольные точки, что является полезным для построения некоторой траектории движения.

Таблица 3.7

Параметры для интерполяции сплайном Катмулла-Рома

X	Y
3	1.26
4	-1.09
4.7	-2
7	1.88
7.8	2
9	1.65

Рис. 3.20. Интерполяция сплайном Катмулла-Рома

На рисунке 3.21 показан график, на котором изображена интерполяция функции полиномом Ньютона по параметрам, заданным в таблице 3.8. Если, при построении методом Лагранжа, приходится пересчитывать всю функцию при добавлении еще одного узла, то алгоритм полинома Ньютона позволяет не производить пересчет всех коэффициентов, кроме последнего.

Таблица 3.8

Параметры для интерполяции полиномом Ньютона

X	Y
8	22
4	-10
11	-20
23	-14
17	-18
10	-3
1	19
6	3
14	20

Рис. 3.21. Интерполяция полиномом Ньютона

ЗАКЛЮЧЕНИЕ

В ходе работы были выполненные поставленные цели и задачи:

- был проведен обзор существующих методов решения задач интерполяции и существующих инструментальных средств для их решения, выводом которого стало отсутствие таковых в бюджетном секторе;
- были реализованы алгоритмы интерполяции, а также, функциональный алгоритм программы;
- был разработан интерфейс приложения и его исходный код.

Результатом работы является разработанное приложение, способное осуществлять интерполяцию различными видами. Оно не является совершенным в плане функционала, но базовые функции выполняет.

На данном этапе приложение умеет выполнять 10 видов интерполяции, что и являлось изначальной целью работы. Исходя из затраченных усилий на разработку программы, можно сказать, что экономическая составляющая проекта себя полностью оправдала, так как затрат получилось намного меньше, чем, например, если бы приобретался уже готовый продукт.

Также, стоит отметить, простоту интерфейса разработанного приложения: пользователю необходимо ввести лишь несколько значений и нажать на несколько кнопок, что упрощает, а, соответственно, ускоряет его работу.

По итогам тестирования разработанного инструментально средства, можно сделать следующий вывод – приложение полностью удовлетворяет всем требования, предъявленным к нему на этапе постановки задачи.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Асташкин, С.В. Интерполяция операторов и ее приложения / С.В. Асташкин. - Москва: Книга по Требованию, 2013. - 188 с.
2. Калиткин, Н.Н. Численные методы / Н.Н. Калиткин Численные методы. –Москва: Наука, 1978.
3. Крейн, С.Г. Интерполяция линейных операторов / С.Г. Крейн, Ю.И. Петунин, Е.М. Семенов. - Москва: Главная редакция физико-математической литературы издательства "Наука", 2014. - 400 с.
4. Половко, А.М. Интерполяция / А.М. Половко. - Москва: Книга по Требованию, 2014. - 313 с.
5. Половко, А.М. Интерполяция. Методы и компьютерные технологии их реализации / А.М. Половко, П.Н. Бутусов. - Москва: БХВ-Петербург, 2016. - 320 с.
6. Рассел, Джесси Бикубическая интерполяция / Д. Рассел. - Москва: Книга по Требованию, 2013. - 814 с.
7. Тюкачев, Н.К. Программирование графики в Delphi / Н.К. Тюкачев, И.Б. Илларионов, В.Н. Хлебостроев. – Москва: Наука, 2008. – 334 с.
8. Хуа, Ло-кен Гармонический анализ функций многих комплексных переменных в классических областях / Ло-кен Хуа. - Москва: [не указано], 2009. - 281 с.
9. Шилов, Г.Е. Математический анализ (функции нескольких вещественных переменных) / Г.Е. Шилов. - М.: Наука, 2010. - 880 с.
10. Тюрин, Ю.Н. Анализ данных на компьютере / Ю.Н. Тюрин, А.А. Макаров. – Москва: Финансы и статистика, 1995.
11. Трибель Х. Теория интерполяции, функциональные пространства, дифференциальные операторы / Х. Трибель. - Москва: Мир, 1980. - 664с.

12. Айфичер, Э.С., Цифровая обработка сигналов: практический подход / Э.С. Айфичер, Б.У. Джервис. - Москва: Издательский дом «Вильяме», 2004.

13. Воеводин, В. В., Матрицы и вычисления / В.В. Воеводин, Ю.А. Кузнецов. - М.: Наука, 1984.

14. Гайдышев, И.И. Анализ и обработка данных: специальный справочник / И.И. Гайдышев. - СПб.: Питер, 2001.

15. Волков, Е.А. Численные методы: учеб. пособие / Е. А. Волков. - Москва: Наука, 1982. - 256 с.

16. Турчак, Л.И. Основы численных методов: учеб. пособие / Л.И. Турчак; под ред. В. В. Щенникова. - Мосвка: Наука, 1987. - 320 с.

17. Поршнев, С.В. Вычислительная математика. Курс лекций: учеб. пособие / С.В. Поршнев. - СПб.: БХВ-Петербург, 2004. - 320 с.

18. Демидович, Б.П. Основы вычислительной математики: учеб. пособие / Б.П. Демидович, И.А. Марон. - СПб.; М.; Краснодар: Лань, 2007. - 672 с.

19. http://vicaref.mgsu.ru/PDE/index12.htm;

20. Бахвалов, Н.С. «Численые методы» / Н.С. Бахвалов - Москва, 2002. - 235 с.

21. http://www.math24.ru/definition-of-fourier-series.html.

22. http://matlab.exponenta.ru/spline/book1/14.php.

23. http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava3.html

ПРИЛОЖЕНИЕ

Листинг 1.1 Реализация методов интерполяции Methods_Int.pas

```
unit Methods_Int;

interface

  uses

  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

  Dialogs, Grids, StdCtrls, Math, Types;

  type

Methods = class

  procedure LinearI(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure Quadr(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure Global(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure LinearSpline(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure QuadrSpline(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure CatmullSpline(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure PolGauss(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

  procedure PolErmita(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
```

```
    procedure Lagranj(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
    procedure Newton(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
  end;

  implementation
  uses MainForm_Int;

  procedure Methods.LinearI(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
    var ColLines, counter: integer;
  begin
    ColLines:=prjMainForm.sgTable.RowCount-1;
    for counter := 1 to ColLines do
prjMainForm.prjGraph_Linear.AddXY(ArrValuesF[counter-1],ArrYValuesF[counter-1]);
  end;

  procedure Methods.Quadr(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
  var
  mat,mata,matb,matc:array[0..2,0..2] of double;
  i,k:integer;
  xnow,ynow,a,b,deter,detera,deterb,deterc,anow,bnow,cnow,j:double;
  begin
```

```
for i := 1 to 15 do
 begin

  for k := 0 to 2 do
  begin
   mat[k,0]:=xar[i+k-1]*xar[i+k-1];
   mat[k,1]:=xar[i+k-1];
   mat[k,2]:=1;
  end;

deter:=(mat[0,0]*mat[1,1]*mat[2,2])+(mat[0,1]*mat[1,2]*mat[2,0])+(mat[1,0]*mat[2,1]*mat[0,
2])
      -(mat[2,0]*mat[1,1]*mat[0,2])-(mat[0,0]*mat[1,2]*mat[2,1])-
(mat[1,0]*mat[0,1]*mat[2,2]);

  for k := 0 to 2 do
  begin
   mata[k,0]:=yar[i+k-1];
   mata[k,1]:=xar[i+k-1];
   mata[k,2]:=1;
  end;
```

```
detera:=(mata[0,0]*mata[1,1]*mata[2,2])+(mata[0,1]*mata[1,2]*mata[2,0])+(mata[1,0]*mata[2,
1]*mata[0,2])

        -(mata[2,0]*mata[1,1]*mata[0,2])-(mata[0,0]*mata[1,2]*mata[2,1])-
(mata[1,0]*mata[0,1]*mata[2,2]);


        for k := 0 to 2 do

          begin

            matb[k,0]:=xar[i+k-1]*xar[i+k-1];

            matb[k,1]:=yar[i+k-1];

            matb[k,2]:=1;

          end;


deterb:=(matb[0,0]*matb[1,1]*matb[2,2])+(matb[0,1]*matb[1,2]*matb[2,0])+(matb[1,0]*matb[2
,1]*matb[0,2])

        -(matb[2,0]*matb[1,1]*matb[0,2])-(matb[0,0]*matb[1,2]*matb[2,1])-
(matb[1,0]*matb[0,1]*matb[2,2]);;


        for k := 0 to 2 do

          begin

            matc[k,0]:=xar[i+k-1]*xar[i+k-1];

            matc[k,1]:=xar[i+k-1];

            matc[k,2]:=yar[i+k-1];
```

```
    end;


deterc:=(matc[0,0]*matc[1,1]*matc[2,2])+(matc[0,1]*matc[1,2]*matc[2,0])+(matc[1,0]*matc[2,
1]*matc[0,2])
        -(matc[2,0]*matc[1,1]*matc[0,2])-(matc[0,0]*matc[1,2]*matc[2,1])-
(matc[1,0]*matc[0,1]*matc[2,2]);


    anow:=detera/deter;
    bnow:=deterb/deter;
    cnow:=deterc/deter;


  if i<15 then
   begin
   j:=0.025;
   xnow:=xar[i];
   while xnow<xar[i+1] do
    begin
     ynow:=a*xnow*xnow+b*xnow+cnow;
     Series1.AddXY(xnow,ynow, '', clBlack);
     xnow:=xar[i] + j;
     j:=j+0.025;
    end;
   end
```

```
        else
        begin
        j:=0.025;
        xnow:=xar[i-1];
        while xnow<xar[i+1] do
         begin
           xnow:=xar[i-1] + j;
           ynow:=anow*xnow*xnow+bnow*xnow+cnow;
           Series1.AddXY(xnow,ynow, '', clBlack);
           j:=j+0.025;
         end;

        end;

      end;
    end;

    procedure Methods.QuadrSpline(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
    var
      C: TRealArray;
      i, j, n: Integer;
      buf, a, b, d, q, r, p1, p2, X: Extended;
```

```
begin
  SetLength(Result, 0);
  If High(PointGrid) < 1 then
    exit;
  C := nc_get_cValues(PointGrid);
  SetLength(c, High(C) + 2);
  c[High(c)] := 0.0;
  n := High(PointGrid);
  X := X1;
  While X < X2 do
    begin
      i := 1;
      While (X > PointGrid[i].X) and (i < n) do
        inc(i);
      j := i - 1;
      a := PointGrid[j].Y;
      b := PointGrid[j].X;
      q := PointGrid[i].X - b;
      r := X - b;
      buf := c[i];
      d := c[i + 1];
      b := (PointGrid[i].Y - a) / q - (d + 2 * buf) * q / 3.0;
      d := (d - buf) / q * r;
```

```
        p1 := b + r * (2 * buf + d);
        p2 := 2 * (buf + d);
        buf := a + r * (b + r * (buf + d / 3.0));
        SetLength(Result, High(Result) + 2);
        Result[High(Result)].X := X;
        Result[High(Result)].Y := buf;
        X := X + Step;
    end;
    end;

    procedure Methods.Global(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
    begin
    a:=xu[i];
    b:=xu[i+1];
    c:=xu[i+2];
    d:=xu[i+3];
    e:=xu[i+4];
    f:=xu[i+5];
    aa:=yu[i];
    bb:=yu[i+1];
    cc:=yu[i+2];
    dd:=yu[i+3];
    ee:=yu[i+4];
```

```
ff:=yu[i+5];

zz:=m;

GlobI:=aa*(((zz-b)*(zz-c)*(zz-d)*(zz-e)*(zz-f))/((a-b)*(a-c)*(a-d)*(a-e)*(a-
f)))+bb*(((zz-a)*(zz-c)*(zz-d)*(zz-e)*(zz-f))/((b-a)*(b-c)*(b-d)*(b-e)*(b-f)))+cc*(((zz-a)*(zz-
b)*(zz-d)*(zz-e)*(zz-f))/((c-a)*(c-b)*(c-d)*(c-e)*(c-f)))+dd*(((zz-a)*(zz-b)*(zz-c)*(zz-e)*(zz-
f))/((d-a)*(d-b)*(d-c)*(d-e)*(d-f)))+ee*(((zz-a)*(zz-b)*(zz-c)*(zz-d)*(zz-f))/((e-a)*(e-b)*(e-
c)*(e-d)*(e-f)))+ff*(((zz-a)*(zz-b)*(zz-c)*(zz-d)*(zz-e))/((f-a)*(f-b)*(f-c)*(f-d)*(f-e)));

end;

procedure Methods.PolGauss(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

var

 i,j,k,l : word;

 s,t : real;begin

 if (n > max) then n := max;

 for i := 1 to n do

  begin

   s := abs(a[i,1]);

   for k := 2 to n do

    begin

     t := abs(a[i,k]);

     if ( t > s) then s := t;

    end;

   for k := 1 to n do a[i,k] := a[i,k]/s;

   b[i] := b[i]/s;

  end;{ ПРЯМОЙ ХОД.}
```

```
for k := 1 to n do
 begin
  j := k;
  s := abs(a[k,k]);
  for i := (k+1) to n do
   begin
    t := abs(a[i,k]);
    if ( t > s) then
     begin
      s := t;
      j := i;
     end
   end;
  if ( j <> k) then
   begin
    for l := k to n do
     begin
      s := a[j,l];
      a[j,l] := a[k,l];
      a[k,l] := s;
     end;
    s := b[j];
    b[j] := b[k];
```

```
      b[k] := s;
     end;
    s := a[k,k];
    a[k,k] := 1.0;
    for l := (k+1) to n do a[k,l] := a[k,l]/s;
    b[k] := b[k]/s;
    for i := (k+1) to n do
     begin
      s := a[i,k];
      a[i,k] := 0.0;
      for l := (k+1) to n do a[i,l] := a[i,l] - a[k,l]*s;
      b[i] := b[i] - b[k]*s
     end;
   end
  for i := (n-1) down to 1 do
    begin
     s := b[i];
     for k := (i+1) to n do s := s - a[i,k]*b[k];
     b[i] := s
    end;
  end;
```

procedure Methods.PolErmita(XValue:real; ArrValuesF:TDoubleDynArray; ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

```
begin
var
   z: Double;
   x1, y1, x2, y2, k1, k2: Double;
   function Cube(x: Double): Double; inline;
   begin
      Result := Sqr(x)*x;
   end;
 begin
   x1 := FP1.x;
   x2 := FP2.x;
   y1 := FP1.y;
   y2 := FP2.y;
   k1 := FK1;
   k2 := FK2;
   Z := Cube(x1 - x2);
   FCoeff.A := (x1*k1-2*y1-x2*k1-x2*k2+x1*k2+2*y2)/Z;
   //((FK1 + FK2) * (FP1.x - Fp2.x ) + 2 * (FP1.y + FP2.y)) / Z;
     FCoeff.B  :=  -(2*Sqr(x1)*k2+Sqr(x1)*k1+3*x1*y2-x1*x2*k2-3*x1*y1+x1*x2*k1-
2*sqr(x2)*k1+3*x2*y2-sqr(x2)*k2-3*x2*y1)/Z;

   {-( FK1 * (Sqr(Fp1.x) + FP1.x * FP2.x - 2 * Sqr(FP2.x)) +  FK2 * (2 *
Sqr(Fp1.x) - FP1.x * FP2.x - Sqr(FP2.x)) +  3 * (FP1.x + FP2.x) * (FP2.y - FP1.y) ) / Z;}
```

```
FCoeff.C := (Cube(x1)*k2+Sqr(x1)*x2*k2+2*Sqr(x1)*x2*k1-2*Sqr(x2)*x1*k2+6*x1*x2*y2-Sqr(x2)*x1*k1-6*x1*x2*y1-Cube(x2)*k1)/Z;

{( Sqr(Fp1.x) * (Fk2 * (FP1.x + FP2.x) + 2 * FP2.x * FK1) -Sqr(FP2.x)* (FK1 * (FP1.x + FP2.x) + 2 * FP1.x * FK2) + 6 * FP1.x * FP2.x * (FP2.y - FP1.y) ) / Z;}

FCoeff.D := -(-Cube(x1)*y2+Cube(x1)*x2*k2+Sqr(x1)*Sqr(x2)*k1-Sqr(x2)*k2*Sqr(x1)+3*x2*y2*Sqr(x1)-3*Sqr(x2)*y1*x1-x1*Cube(x2)*k1+Cube(x2)*y1)/Z;

{-( 3 * FP1.x * FP2.x * (FP2.y * FP1.x - FP2.x + FP1.y) + Fp1.x * FP2.x * (FP1.x - FP2.x) * (FK1 * FP2.x + Fk2 + FP1.x) - IntPower(FP1.x, 3) * FP2.y + IntPower(Fp2.x, 3) * FP1.y ) / Z;}

  if Abs(GetValueIn(x1) - y1) > 1e-6 then
    Assert(False);
  if Abs(GetValueIn(x2) - y2) > 1e-6 then
    Assert(False);
 end;
//Значение
result := FCoeff.A * IntPower(APoint, 3) + FCoeff.B * Sqr(APoint) + FCoeff.C * APoint + FCoeff.D;

end;
```

```
procedure Methods.Lagranj(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);

var

  i, j, n: Integer;

  _1: Extended;

  l1, l2: TRealArray;

begin

  n := High(Points);

  SetLength(l1, n + 1);

  SetLength(l2, n + 1);

  for i := 0 to n do

    l1[i] := Points[i].Y;

  for j := 1 to n do

    begin

      for i := 0 to n - j do

        begin

          _1 := Points[j + i].X - Points[i].X;

          If _1 < 0.001 then

            _1 := 0.001;

          l2[i] := (l1[i] * (Points[i + j].X - X) - l1[i + 1] * (Points[i].X - X)) /

                (_1);

        end;

      SetLength(l2, High(l2));

      l1 := l2;
```

```
    end;
  Result := l1[0];
end;

procedure Methods.Newton(XValue:real; ArrValuesF:TDoubleDynArray;
ArrYValuesF:TDoubleDynArray; Calculate: Boolean);
var n: Integer;
  function item(idx: Integer):Double;
  var
    tmp: Double;
    i: Integer;
  begin
    if idx = 0 then
      Result = y[idx]
    else
      begin
        tmp := Math.IntPower(delta(y[n-idx]), idx);
        for I := n downto n-idx do
          tmp := tmp * (x_point - x[i]);
        result := tmp / factorial(idx) / Math.IntPower(h, idx);
      end;
  end;var
  i: Integer;
  sum: double;begin
```

```
    n := Length(x);
    sum := 0;
    for i := 0 to n do
        sum := sum + item(i);
    result := sum;
end;


end.
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись)

(Ф.И.О.)