Цель работы:

а) освоение методов интерполяции функций;

б) совершенствование навыков по алгоритмизации и программированию вычислительных задач.

Задание:

Составить схему алгоритма и программу на языке C/C++ решения задачи по теме «Интерполяция» в соответствии с индивидуальным заданием.

Вариант 14

Значения x: -1.0, -0.5, 0.0, 0.67, 1.0

Значения y: 1.0, -0.25, 0.0, 0.67, 1.0

Метод интерполяции: метод Лагранжа

Общая постановка задачи:

При решении различных практических задач результаты исследований оформляются в виде таблиц, отображающих зависимость одной или нескольких измеряемых величин от одного определяющего параметра (аргумента). Такого рода таблицы представлены обычно в виде двух или более строк (столбцов) и используются для формирования математических моделей.

Таблично заданные в математических моделях функции обычно записываются в таблицы вида:

X	X_0	X_1	…	X_n
$Y_1(X)$	$Y(X_0)$	$Y(X_1)$	…	$Y(X_n)$
…	…	…	…	…
$Y_m(X)$	$Y(X_0)$	$Y(X_1)$	…	$Y(X_n)$

(1)

Ограниченность информации, представленной такими таблицами, в ряде случаев требует получить значения функций $Y_j(X)$ $(j=1,2,\ldots,m)$ в точках X, не совпадающих с узловыми точками таблицы X_i $(i=0,1,2,\ldots,n)$. В таких случаях необходимо определить некоторое аналитическое выражение $\varphi_j(X)$ для вычисления приближенных значений исследуемой функции $Y_j(X)$ в произвольно задаваемых точках X. Функция $\varphi_j(X)$ используемая для определения приближенных значений функции $Y_j(X)$ называется аппроксимирующей

функцией (от латинского ***approximo*** - приближаюсь). Близость аппроксимирующей функции $\varphi_j(X)$ к аппроксимируемой функции $Y_j(X)$ обеспечивается выбором соответствующего алгоритма аппроксимации.

Все дальнейшие рассмотрения и выводы мы будем делать для таблиц, содержащих исходные данные одной исследуемой функции (т. е. для таблиц с $m=1$).

Интерполяция каноническим полиномом

Метод интерполяции функции каноническим полиномом основывается на построении интерполирующей функции как полинома в виде [1]

$$\varphi(x) = P_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n \qquad (4)$$

Коэффициенты c_i полинома (4) являются свободными параметрами интерполяции, которые определяются из условий Лагранжа:

$$P_n(x_i) = Y_i, \quad (i = 0, 1, \ldots, n) \qquad (5)$$

Используя (4) и (5) запишем систему уравнений

$$\begin{aligned} &c_0 + c_1 x_0 + c_2 x_0^2 + \ldots + c_n x_0^n = Y_0 \\ &c_0 + c_1 x_1 + c_2 x_1^2 + \ldots + c_n x_1^n = Y_1 \\ &\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ &c_0 + c_1 x_n + c_2 x_n^2 + \ldots + c_n x_n^n = Y_n \end{aligned} \qquad (6)$$

Вектор решения c_i ($i = 0, 1, 2, \ldots, n$) системы линейных алгебраических уравнений (6) существует и может быть найден, если среди узлов x_i нет совпадающих. Определитель системы (6) называется определителем Вандермонда[1] и имеет аналитическое выражение [2].

Для определения значений коэффициентов c_i (i = 0, 1, 2, … , n) систему уравнений (5) можно записать в векторно-матричной форме

---

[1] *Определителем Вандермонда* называется определитель

$$\begin{vmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & \ldots & x_n^{n-1} \end{vmatrix} = \prod_{1 \le j < i \le n} (x_i - x_j) = (x_2 - x_1)(x_3 - x_2)(x_4 - x_3)\ldots(x_{n+1} - x_n)$$

$$\prod_{k=1}^{n} x_k = x_1 x_2 x_3 \ldots x_n$$

.

Он равен нулю тогда и только тогда, когда $x_i = x_j$ для некоторых $i \neq j$. (Материал из Википедии — свободной энциклопедии)

$$A * C = \bar{Y}\ , \qquad (7)$$

где *A,* матрица коэффициентов, определяемых таблицей степеней вектора аргументов $X= (x_i^0, x_i, x_i^2, \dots , x_i^n)^T$ $(i = 0, 1, 2, \dots , n)$

$$A = \begin{Vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ . & . & . & . & . \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{Vmatrix}, \qquad (8)$$

$\boldsymbol{C}$ - вектор-столбец коэффициентов c_i (i = 0, 1, 2, … , n), а $\bar{Y}$ - вектор-столбец значений Y_i $(i = 0, 1, 2, \dots , n)$ интерполируемой функции в узлах интерполяции.

Интерполяционная формула Лагранжа

Предложенный Лагранжем алгоритм построения интерполирующих функций по таблицам (1) предусматривает построение интерполяционного многочлена $L_n(x)$ в виде

$$L_n(x) = l_0(x) + l_1(x) + \dots + l_n(x) \qquad (11)$$

где $l_i(x)$ - многочлен степени n, для которого выполняются условия

$$l_i(x_k) = \begin{cases} Y_i, \text{ если } i = k, \\ 0, \text{ если } i \neq k. \end{cases} \qquad (12)$$

Очевидно, что выполнение для (10) условий (11) определяет выполнение условий (2) постановки задачи интерполяции.

Многочлены $l_i(x)$ записываются следующим образом

$$l_i(x) = q_i(x-x_0)(x-x_1)\cdot\dots\cdot(x-x_{i-1})(x-x_{i+1}) \ \dots \ (x-x_n). \qquad (13)$$

Здесь q_j - константа, значение которой определяется с учётом (12) как

$$q_i = \frac{Y_i}{(x_i - x_0)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}, \ (i = 0,1,2,\dots,n) \qquad (14)$$

Отметим, что ни один множитель в знаменателе формулы (14) не равен нулю. Вычислив значения констант c_i, можно использовать их для вычисления значений интерполируемой функции в заданных точках.

Формула интерполяционного многочлена Лагранжа (11) с учётом формул (13) и (14) может быть записана в виде

$$L_n(x) = \sum_{i=0}^{n} q_i (x - x_0)(x - x_1) \bullet \ldots \bullet (x - x_{i-1})(x - x_{i+1}) \bullet \ldots \bullet (x - x_n) \qquad (15)$$

Аналитические расчеты:

Расчеты в программе Excel приведены на рисунке 1. Красным отмечены точки экстраполяции (вне исходного интервала), зеленым - интерполяции.

№	X	Y	q_i	$l_0(X)$	$l_1(X)$	$l_2(X)$	$l_3(X)$	$l_4(X)$	Yрасч(X)
0	-1,00	1,00	0,599	1,000	0,000	0,000	0,000	0,000	1,000
1	-0,50	-0,25	0,570	0,000	-0,250	0,000	0,000	0,000	-0,250
2	0,00	0,00	0,000	0,000	0,000	0,000	0,000	0,000	0,000
3	0,67	0,67	-1,551	0,000	0,000	0,000	0,670	0,000	0,670
4	1,00	1,00	1,010	0,000	0,000	0,000	0,000	1,000	1,000
	-1,4			3,748	1,585	0,000	-1,876	1,054	4,511
	-0,8			0,380	-0,241	0,000	0,134	-0,071	0,202
	-0,3			-0,045	-0,151	0,000	-0,085	0,041	-0,240
	0,5			0,025	0,036	0,000	0,582	-0,129	0,515
	0,8			-0,016	-0,021	0,000	0,581	0,246	0,789
	1,5			0,746	0,887	0,000	-5,816	6,288	2,104

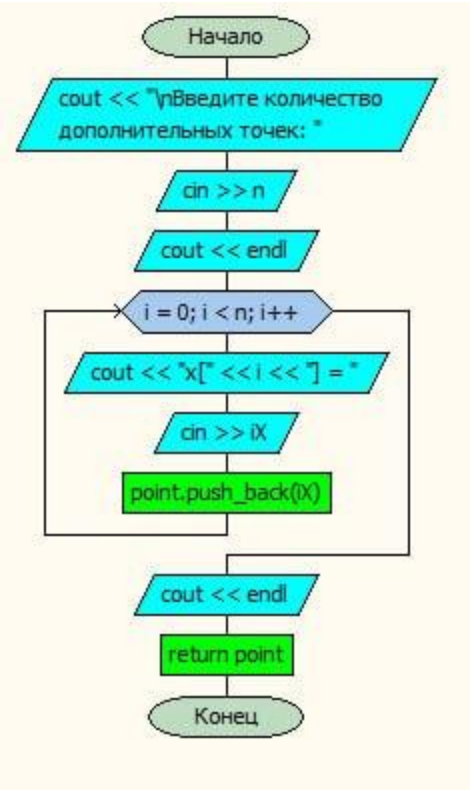
Рисунок 1– Результат расчетов в программе Excel

X	Y(X)	Yрасч(X)
-1,4		4,511
-1	1,000	1,000
-0,8		0,202
-0,5	-0,250	-0,250
-0,3		-0,240
0	0,000	0,000
0,5		0,515
0,67	0,670	0,670
0,8		0,789
1	1,000	1,000
1,5		2,104

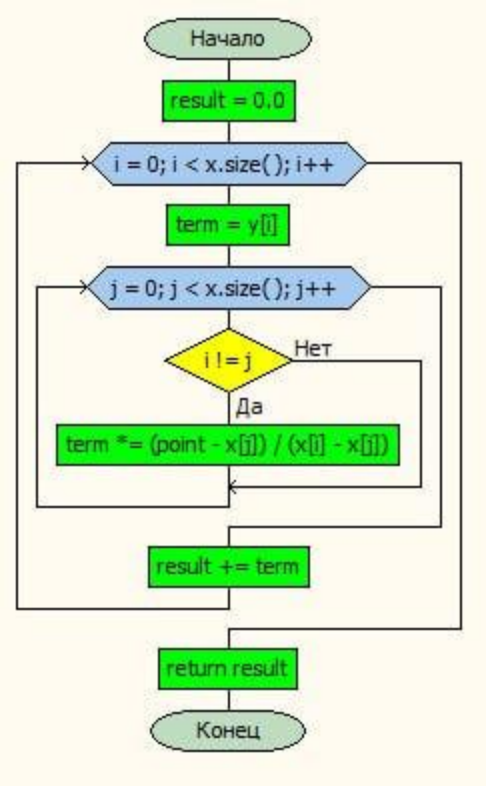
Рисунок 2. Таблица и диаграмма, отображающая результаты интерполяции по Лагранжу

Блок схемы:

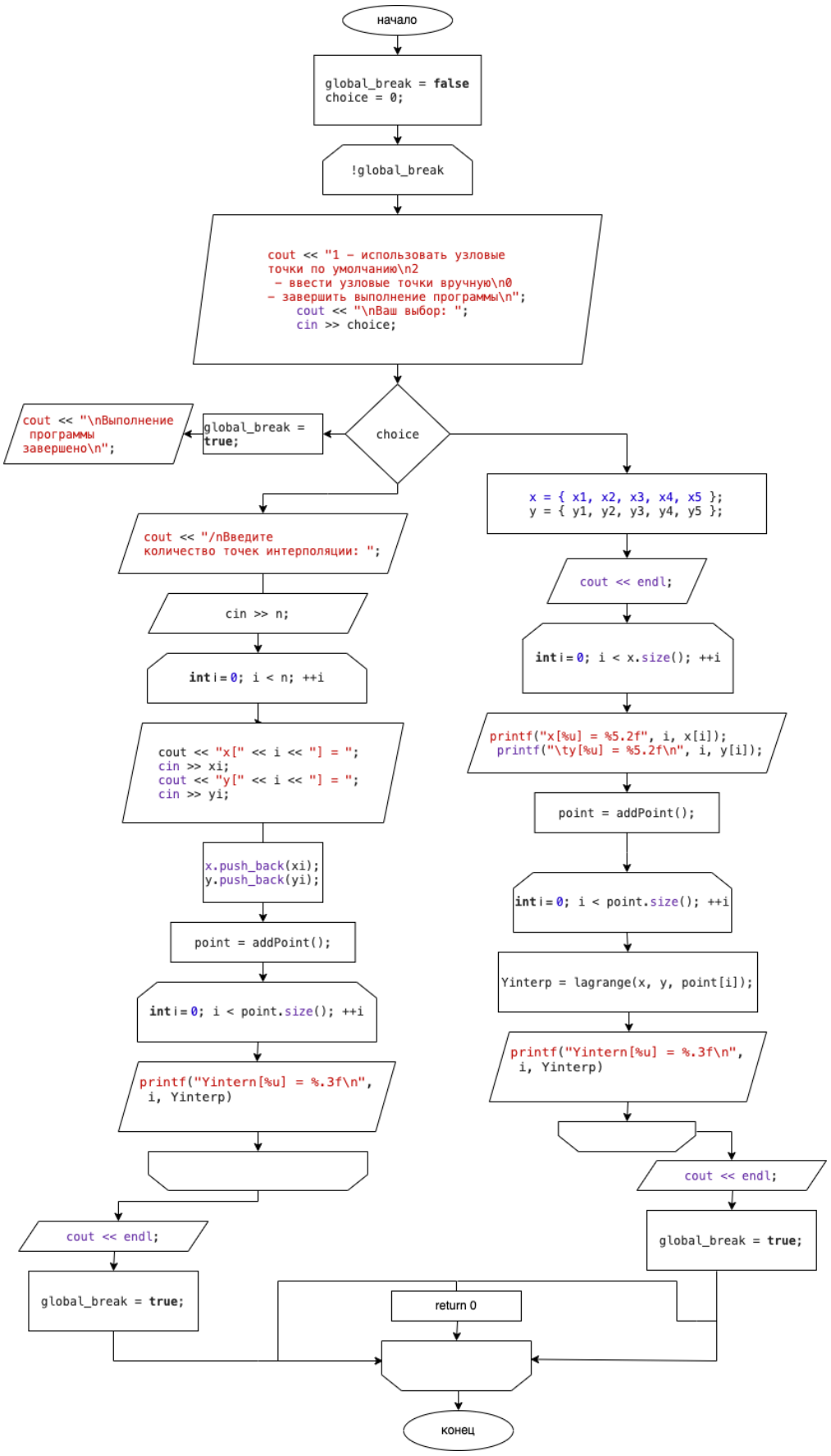
addPoint():



Lagrange():



main():


начало
global_break = false
choice = 0;
!global_break
cout << "1 – использовать узловые
точки по умолчанию\n2
– ввести узловые точки вручную\n0
– завершить выполнение программы\n";
cout << "\nВаш выбор: ";
cin >> choice;
choice
global_break =
true;
cout << "\nВыполнение
программы
завершено\n";
cout << "/nВведите
количество точек интерполяции: ";
cin >> n;
int i = 0; i < n; ++i
cout << "x[" << i << "] = ";
cin >> xi;
cout << "y[" << i << "] = ";
cin >> yi;
x.push_back(xi);
y.push_back(yi);
point = addPoint();
int i = 0; i < point.size(); ++i
printf("Yintern[%u] = %.3f\n",
i, Yinterp)
cout << endl;
global_break = true;
x = { x1, x2, x3, x4, x5 };
y = { y1, y2, y3, y4, y5 };
cout << endl;
int i = 0; i < x.size(); ++i
printf("x[%u] = %5.2f", i, x[i]);
printf("\ty[%u] = %5.2f\n", i, y[i]);
point = addPoint();
int i = 0; i < point.size(); ++i
Yinterp = lagrange(x, y, point[i]);
printf("Yintern[%u] = %.3f\n",
i, Yinterp)
cout << endl;
global_break = true;
return 0
конец

Листинг кода программы:

```
#include <iostream>
#include <vector>

using namespace std;

vector<double> addPoint() {
    int n;
    double iX;
    vector<double> point;

    cout << "\nВведите количество дополнительных точек: ";
    cin >> n;
    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << "x[" << i << "] = ";
        cin >> iX;
        point.push_back(iX);
    }
    cout << endl;
    return point;
}

double lagrange(vector<double> x, vector<double> y, double point) {
    double result = 0.0;

    for (int i = 0; i < x.size(); i++) {
        double term = y[i];

        for (int j = 0; j < x.size(); j++) {
            if (i != j) {
                term *= (point - x[j]) / (x[i] - x[j]);
            }
        }
        result += term;
    }
    return result;
}

int main() {
    bool global_break = false;
    int choice = 0;

    while (!global_break) {
        cout << "1 - использовать узловые точки по умолчанию\n2 - ввести узловые точки вручную\n0 -
завершить выполнение программы\n";
        cout << "\nВаш выбор: ";
        cin >> choice;

        switch (choice){
            case 0:
                global_break = true;
                cout << "\nВыполнение программы завершено\n";
                break;

            case 1: {
                vector<double> x = {-1.0, -0.5, 0.0, 0.67, 1.0};
                vector<double> y = {1.0, -0.25, 0.0, 0.67, 1.0};

                cout << endl;
                for (int i = 0; i < x.size(); ++i) {
```

```
            printf("x[%u] = %5.2f", i, x[i]);
            printf("\ty[%u] = %5.2f\n", i, y[i]);
        }

        vector<double> point = addPoint();

        for (int i = 0; i < point.size(); ++i) {
            double Yinterp;
            Yinterp = lagrange(x, y, point[i]);
            printf("Yintern[%u] = %.3f\n", i, Yinterp);
        }

        cout << endl;
        global_break = true;
        break;
      }
      case 2: {
        int n;
        vector<double> x;
        vector<double> y;

        cout << "/nВведите количество точек интерполяции: ";
        cin >> n;

        for (int i = 0; i < n; ++i) {
            double xi, yi;
            cout << "x[" << i << "] = ";
            cin >> xi;
            cout << "y[" << i << "] = ";
            cin >> yi;
            x.push_back(xi);
            y.push_back(yi);
        }

        vector<double> point = addPoint();

        for (int i = 0; i < point.size(); ++i) {
            double Yinterp;
            Yinterp = lagrange(x, y, point[i]);
            printf("Yintern[%u] = %.3\n", i, Yinterp);
        }
        cout << endl;
        global_break = true;
        break;
      }
      default:
        cout << "\nТакого действия нет. Повторите ввод:\n\n";
        break;
    }
  }
  return 0;
}
```

Результат работы программы:

1 – используются данные из аналитических расчетов:

```
1 - использовать узловые точки по умолчанию
2 - ввести узловые точки вручную
0 - завершить выполнение программы

Ваш выбор: 1

x[0] = -1.00    y[0] =  1.00
x[1] = -0.50    y[1] = -0.25
x[2] =  0.00    y[2] =  0.00
x[3] =  0.67    y[3] =  0.67
x[4] =  1.00    y[4] =  1.00

Введите количество дополнительных точек: 6

x[0] = -1.4
x[1] = -0.8
x[2] = -0.3
x[3] = 0.5
x[4] = 0.8
x[5] = 1.5

Yintern[0] = 4.511
Yintern[1] = 0.202
Yintern[2] = -0.240
Yintern[3] = 0.515
Yintern[4] = 0.789
Yintern[5] = 2.104
```

Рис 3. Данные из аналит. расчетов

2 – ручной ввод:

```
1 - использовать узловые точки по умолчанию
2 - ввести узловые точки вручную
0 - завершить выполнение программы

Ваш выбор: 2

Введите количество точек интерполяции: 5

x[0] = -1
y[0] = 1

x[1] = -0.5
y[1] = -0.25

x[2] = 0
y[2] = 0

x[3] = 0.67
y[3] = 0.67

x[4] = 1
y[4] = 1
```

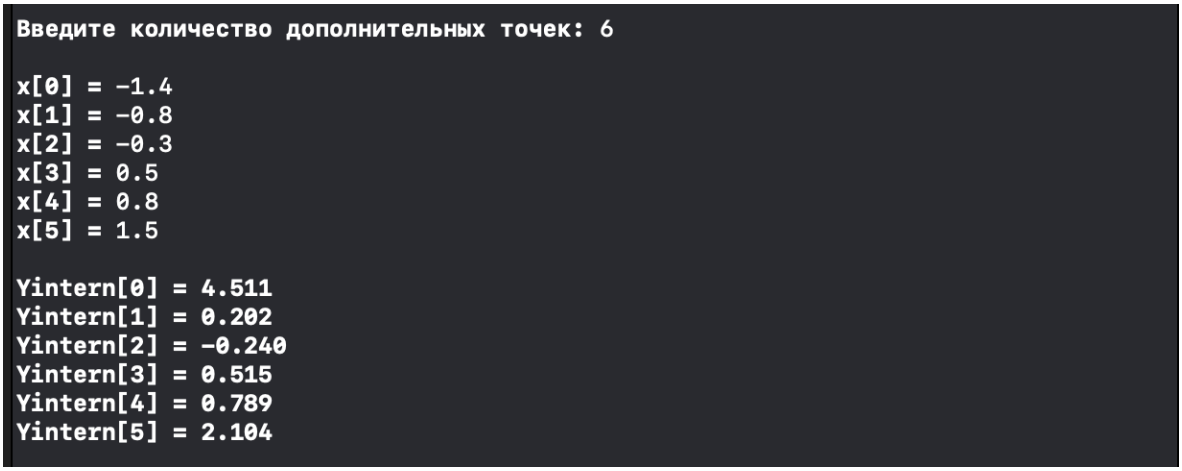


Рис 4. Ручной ввод

Сравнение результатов аналитического и программного расчета:

	аналитический	программный
X = -1.4	4.511	4.511
X = -0.8	0.202	0.202
X = -0.3	-0.24	-0.24
X = 0.5	0.515	0.515
X = 0.8	0.789	0.789
X = 1.5	2.104	2.104

Исходя из данных таблицы (все данные совпадают), мы можем сделать вывод, что программа работает верно.

Вывод:

В ходе выполнения практической работы №3 была написана программа, которая решает задачу интерполяции функции используя при этом метод Лагранжа. Результат программного расчета при одинаковых входных данных полностью совпал с

результатом аналитического расчета, что говорит о том, что программа работает корректно и выдает правильные результаты.