Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Петрозаводский государственный университет»  
Физико-технический институт  
Направление Информатика и вычислительная техника. Проектирование информационных систем в экономике

ОТЧЁТ  
по лабораторной работе №3  
**«Метод наименьших квадратов»**

Автор работы:

студентка группы 21218

Э.В. Таничева

«27» октября 2022 г.

**Формулировка задачи: Вариант 13**

На языке Программирования высокого уровня составить программу, аппроксимирующую предложенный набор экспериментальных точек полиномом по методу наименьших квадратов. Обосновать, какая степень полинома необходима для правильной аппроксимации (абсолютная погрешность исходных данных +-1.5). Решить задачу аппроксимации с помощью любой экспертной программы. Построить график полученного полинома, аппроксимирующего экспериментальные значения.

**X: 1 2.2 2.4 2.7 3.1 3.5 4.5 5**

**Y: 1.009 -11.821 -14.827 -20.02**

**-29.099 -36.57 -42.577 -31.651**

**Сведения о численном методе, которым будет решаться задача:**

**Постановка задачи аппроксимации по МНК. Условия наилучшего приближения.**

Если набор экспериментальных данных получен со значительной погрешностью, то интерполяция не только не требуется, но и нежелательна! Здесь требуется построить кривую, которая воспроизводила бы график исходной экспериментальной закономерности, т.е. была бы максимально близка к экспериментальным точкам, но в то же время была бы нечувствительна к случайным отклонениям измеряемой величины.

Введем непрерывную функцию *φ*(*x*) для аппроксимации дискретной зависимости *f*(*x*i), i = 0…*n*. Будем считать, что *φ*(*x*) построена по условию *наилучшего квадратичного приближения*, если

. (1)

Весу *ρ* для *i*-й точки придают смысл точности измерения данного значения: чем больше *ρ*, тем ближе аппроксимирующая кривая «притягивается» к данной точке. В дальнейшем будем по умолчанию полагать *ρ* = 1 для всех точек.

Рассмотрим случай *линейной аппроксимации*:

*φ*(*x*) = *c*0*φ*0(*x*) + *c*1*φ*1(*x*) + … + *c*m*φ*m(*x*), (2)

где *φ*0…*φ*m ­– произвольные *базисные функции*, *c*0…*c*m – неизвестные коэффициенты, *m* < *n*. Если число коэффициентов аппроксимации взять равным числу узлов, то среднеквадратичная аппроксимация совпадет с интерполяцией Лагранжа, при этом, если не учитывать вычислительную погрешность, *Q* = 0.

Если известна экспериментальная (исходная) погрешность данных *ξ*, то выбор числа коэффициентов, то есть величины *m*, определяется условием:

. (3)

Иными словами, если , число коэффициентов аппроксимации недостаточно для правильного воспроизведения графика экспериментальной зависимости. Если , многие коэффициенты в (2) не будут иметь физического смысла.

Для решения задачи линейной аппроксимации в общем случае следует найти условия минимума суммы квадратов отклонений для (2). Задачу на поиск минимума можно свести к задаче поиска корня системы уравнений , *k* = 0…*m*. (4).

Подстановка (2) в (1), а затем расчет (4) приведет в итоге к следующей системе *линейных* *алгебраических* уравнений:



Далее следует решить полученную СЛАУ относительно коэффициентов *c*0…*c*m. Для решения СЛАУ обычно составляется расширенная матрица коэффициентов, которую называют *матрицей Грама*, элементами которой являются скалярные произведения базисных функций и столбец свободных коэффициентов:

,

где , , *j* = 0…*m*, *k* = 0…*m*.

После того как с помощью, например, метода Гаусса найдены коэффициенты *c*0…*c*m, можно построить аппроксимирующую кривую или вычислить координаты заданной точки. Таким образом, задача аппроксимации решена.

**Аппроксимация каноническим полиномом.**

Выберем базисные функции в виде последовательности степеней аргумента x:

*φ*0(*x*) = *x*0 = 1; *φ*1(*x*) = *x*1 = *x*; *φ*m(*x*) = *x*m, *m* < *n*.

Расширенная матрица Грама для степенного базиса будет выглядеть следующим образом:

.

Особенность вычислений такой матрицы (для уменьшения количества выполняемых действий) состоит в том, что необходимо сосчитать только элементы первой строки и двух последних столбцов: остальные элементы заполняются сдвигом предшествующей строки (за исключением двух последних столбцов) на одну позицию влево. В некоторых языках программирования, где отсутствует быстрая процедура возведения в степень, пригодится алгоритм расчета матрицы Грама, представленный далее.

Выбор базисных функций в виде степеней *x* *не является оптимальным* с точки зрения достижения наименьшей погрешности. Это является следствием **неортогональности** выбранных базисных функций. Свойство *ортогональности* заключается в том, что для каждого типа полинома существует отрезок [*x*0, *x*n], на котором обращаются в нуль скалярные произведения полиномов разного порядка:

, *j* ≠ *k*, *ρ* ­– некоторая весовая функция.

Если бы базисные функции были ортогональны, то все недиагональные элементы матрицы Грама были бы близки к нулю, что увеличило бы точность вычислений, в противном случае при определитель матрицы Грама очень быстро стремится к нулю, т.е. система становится плохо обусловленной.

**Блок-схема:**

НАЧАЛО

КОНЕЦ

M – степень полинома,

набор значений (x**i**, f**i**), где i = 0…N

Вывод y;

вывод погрешности

и набора коэффициентов для аппроксимации

Q**i**=1;

i = 0…N

S = 0;

R = 0;

j = 0

i = 0

j = j + 1

j > N

ДА

A**0 i** = S

A**i М+1** = R

НЕТ

Решение СЛАУ

методом Гаусса

Формирование верхней строки матрицы Грама и последнего столбца

S = S + Q**j**

R = R + Q**j** \* f**j**

Q**j** = Q**j** \* x**j**

i = i + 1

i > M

НЕТ

ДА

i = 1

S = 0;

j = 0

j < M

A**ij** = A **i-1 j+1**

S = S + Q**j**

Q**j** = Q**j** \* x**j**

ДА

НЕТ

j = j + 1

j > N

A**i M** = S;

i = i + 1

НЕТ

ДА

i > M

ДА

НЕТ

В массиве Q будут храниться текущие значения степеней узлов

Массив А размера [0…m, 0…m+1] будет содержать сформированную матрицу

В S накапливаются значения для каждого элемента матрицы Грама

R накапливает значения для последнего столбца

Формирование оставшихся строк матрицы Грама

**Листинг программы:**

**Program** lab3;

**const**

M=4;

**var**

n,n1,k,i,j: integer;

x,x1: **array**[0..100] **of** real; //x - расчеты С коэфф-ов, х1 - изначальные эл-ты х

f: **array**[0..100] **of** real; //f - изначальный эл-ты у, у - расчеты

y: **array**[0..100] **of** real;

Q: **array**[0..100] **of** real; //значение степеней узлов

A: **array**[0..100, 0..100] **of** real;

A1: **array**[1..100, 1..100] **of** real; //разбиение матрицы А[i,j] - матрицы коэфф-ов

b: **array**[1..100] **of** real; //разбиение матрицы А[i,j] - свободные члены

R,q1,c,s: real;

**BEGIN**

writeln(' Введите кол-во узлов (кол-во пар x и у, учитывая, что подсчет идет с 0): '); //Вводим значения кол-ва узлов и координат х,у

write(' ');readln(n);

writeln;

writeln(' Введите все значения х: ');

**for** i:=0 **to** n **do**

**begin**

write(' x',i,'= '); readln(x[i]);

x1[i]:=x[i];

**end**;

i:=0;

writeln;

writeln(' Введите все значения у: ');

**for** i:=0 **to** n **do**

**begin**

write(' y',i,'= '); readln(f[i]);

**end**;

**for** i := 0 **to** N **do**

Q[i] := 1;

i := 0;

**repeat**

S := 0;

R := 0; //накапливает значения для последнего столбца

j := 0;

**repeat**

S := S + Q[j];

R := R + Q[j] \* f[j];

Q[j] := Q[j] \* x[j];

j := j + 1;

**until** j > N;

A[0, i] := S; //заполнение первой строки

A[i, M + 1] := R; //содержит сформатированную матрицу

i := i + 1;

**until** i > M;

i := 1;

**repeat**

S := 0;

j := 0;

**repeat**

**if** j < M **then**

**begin**

A[i, j] := A[i - 1, j + 1];

**end**;

S := S + Q[j]; //накапливает значения для каждого элемента матрицы Грама

Q[j] := Q[j] \* x[j];

j := j + 1;

**until** j > N;

A[i, M] := S;

i := i + 1;

**until** i > M;

n1 := m+1;

//разделим матрицу A на матрицу коэффициентов и матрицу свободных членов b

**for** i:= 1 **to** n1 **do**

**for** j:= 1 **to** n **do**

A1[i, j] := A[i-1, j-1];

j:=n1;

**for** i:= 1 **to** n1 **do**

b[i] := A[i-1, j];

//Решение СЛАУ: прямой ход

**for** k := 1 **to** n1 – 1 **do**

**begin**

**for** i := k + 1 **to** n1 **do**

**begin**

c := A1[i, k] / A1[k, k];

A1[i, k] := 0;

**for** j := k + 1 **to** n1 **do**

**begin**

A1[i, j] := A1[i, j] - A1[k, j] \* c;

**end**;

b[i] := b[i] - b[k] \* c;

**end**;

**end**;

x[n1] := b[n1] / A1[n1, n1];

//обратный

**for** i := n1 – 1 **downto** 1 **do**

**begin**

s := 0;

**for** j := i + 1 **to** n1 **do**

**begin**

s := s + A1[i, j] \* x[j];

**end**;

x[i] := (b[i] - s) / A1[i, i];

**end**;

writeln;

writeln(' Набор точек х и у попарно:');

**for** i:=0**to** 7 **do**

**begin**

y[i]:=x[m];

**for** j:= m-1 **downto** 0 **do**

y[i]:= x[j]+x[i]\*y[i];

writeln(' x:= ', x1[i]: 1: 1,' y:= ',f[i]:0:3);

**end**;

writeln;

writeln(' Коэффициенты с: ');

**for** i := 1 **to** n1 **do**

writeln(' c[', i - 1, '] = ', x[i]:0:3);

writeln;

writeln(' Уравнение ',m, '-й степени полинома '); { вывод полинома}

write(' y:= ');

**for** i:=m+1 **downto** 1 **do**

**begin**

write(x[i]:0:3);

**if** i>2 **then**

write('\*(X^',i-1,')');

**if** i=2 **then**

write('\*(X)');

**if** x[i]<0 **then** write(' + ');

**end**;

**for** i:= 0 **to** n **do**

**begin**

y[i]:=x[m+1];

**for** j:= m-1 **downto** 0 **do**

y[i]:= x[j+1]+x1[i]\*y[i];

**end**;

q1:=0;

**for** i:=0 **to** 7 **do**

**begin**

q1:=q1+sqr(f[i]-y[i]);

**end**;

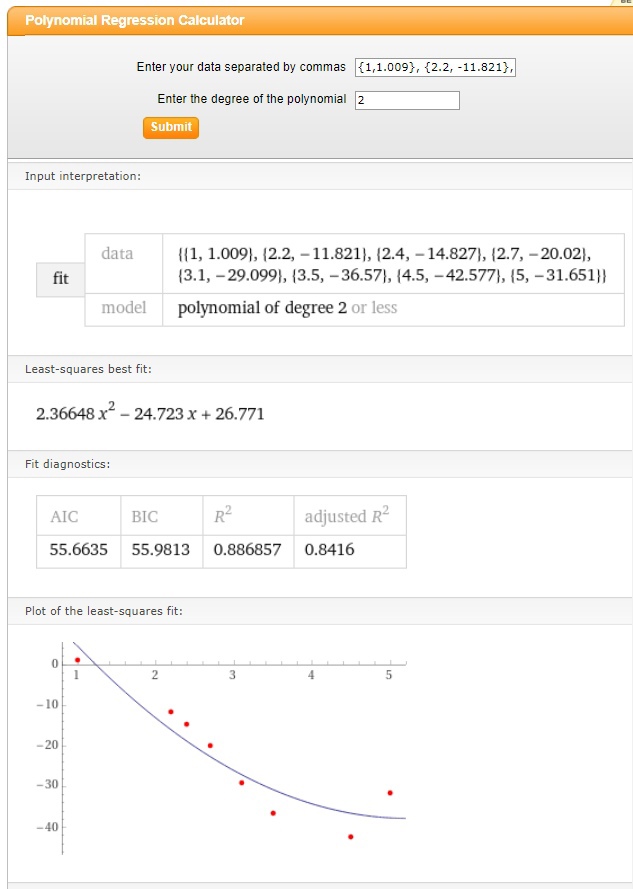
writeln(' ');

writeln(' Q(Погрешность):= ', sqrt(q1):2:2);

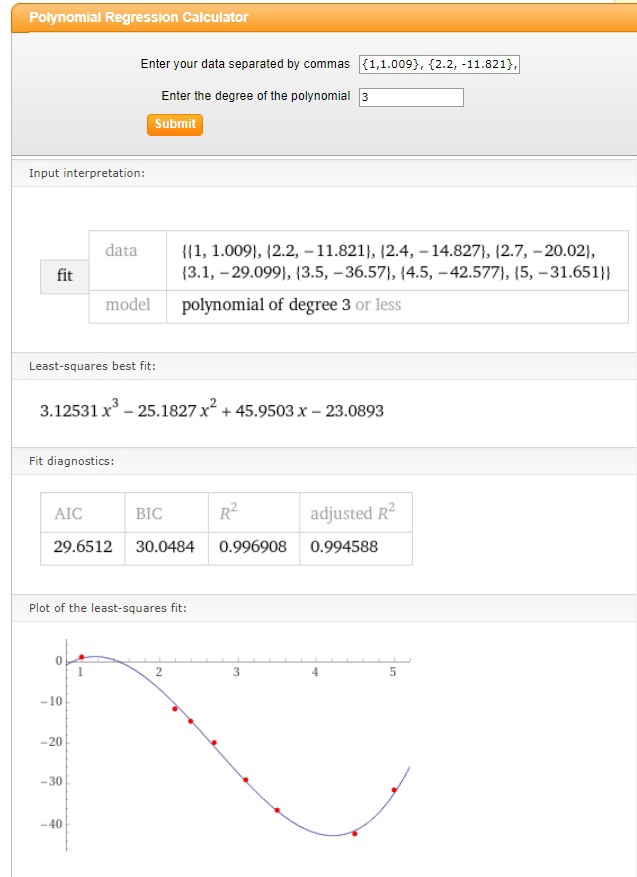
**end**.

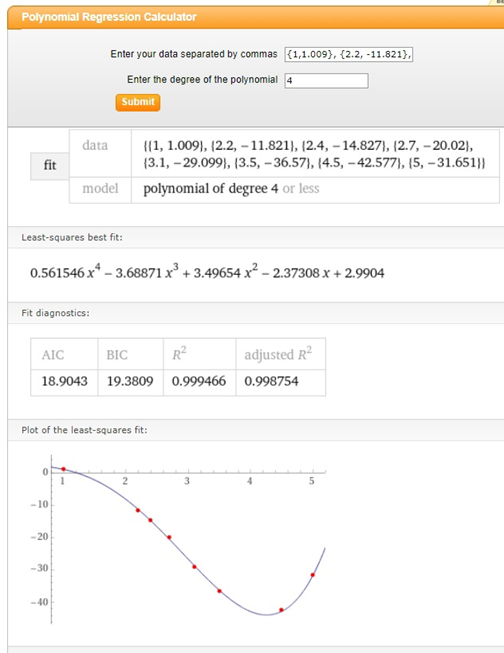
**Результаты расчета в среде WolframAlpha:**

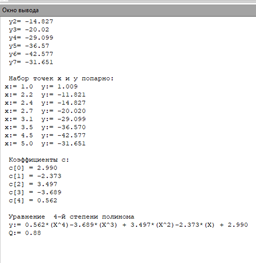
При m=2

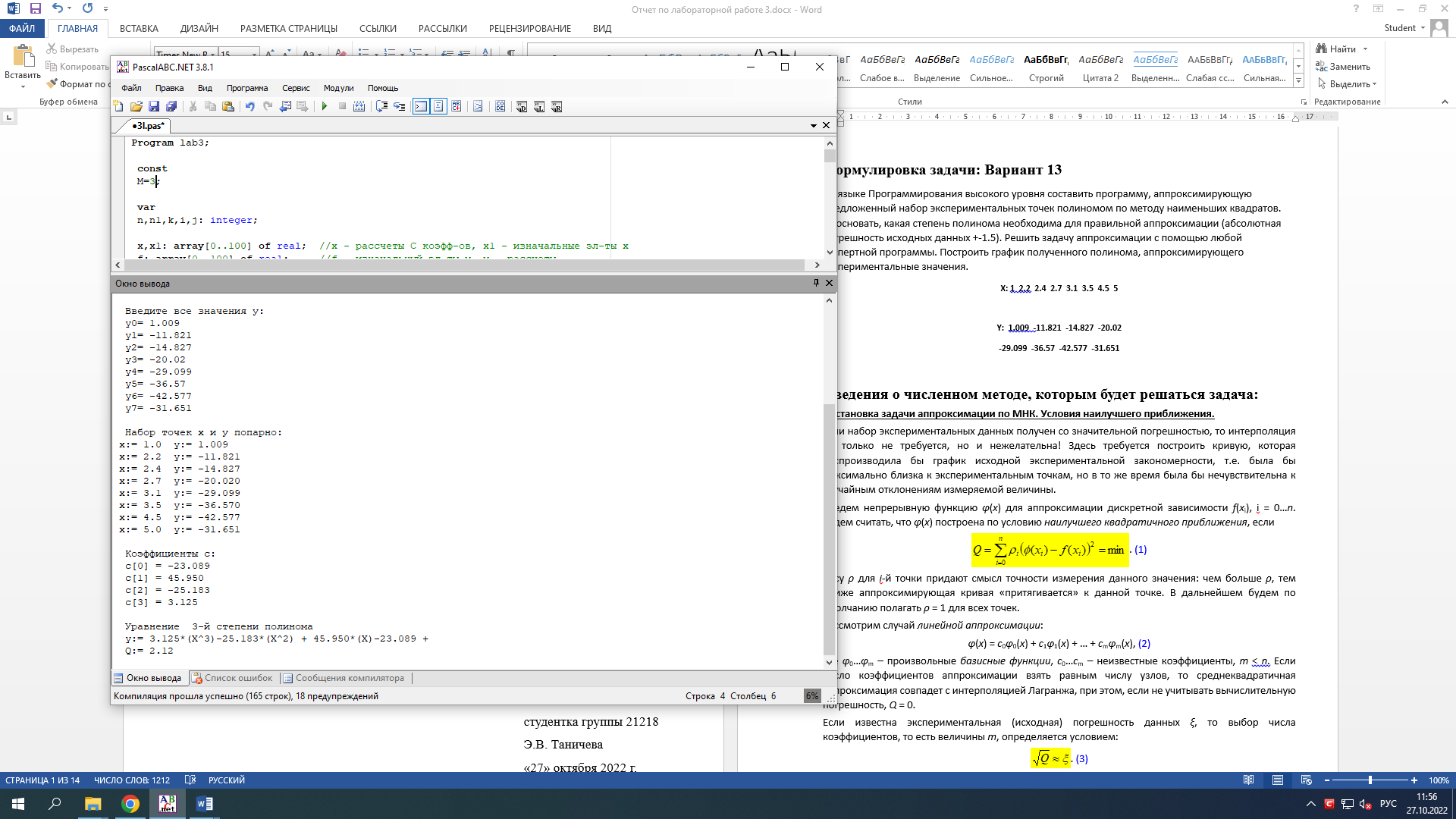


При m=3



При m=4

**Результаты работы программы:** При m=4****

При m=3

**Bывод:**

С помощью программы PascalABC я написала программу, которая считает по методу наименьших квадратов. Ближе всего Q => 1,5 при M=3 и М=4 (у М=2 результат слишком отличается), отсюда можно сделать вывод, что наиболее оптимальной степенью полинома является 3-я степень=2,12 и 4-я степень=0,88.