Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3
Задание №3 по курсу «Численные методы»
Вариант №12

Студент:	С. М. Власова
Преподаватель:	И. Э. Иванов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Задание №3.3

Для таблично заданной функции путем решения нормальной системы МНК найти приближающие многочлены а) 1-ой и б) 2-ой степени. Для каждого из приближающих многочленов вычислить сумму квадратов ошибок. Построить графики приближаемой функции и приближающих многочленов.
Вариант: 12

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
y_i	-1.8415	0.0	1.8415	2.9093	3.1411	3.2432

1 Описание метода решения

МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

Пусть задана таблично в узлах x_j функция $y_j = f(x_j), \quad j = 0, 1, \ldots, N$. При этом значения функции y_j определены с некоторой погрешностью, также из физических соображений известен вид функции, которой должны приближенно удовлетворять табличные точки, например: многочлен степени n, у которого неизвестны коэффициенты a_i, $F_n(x) = \sum_{i=0}^{n} a_i x^i$. Неизвестные коэффициенты будем находить из условия минимума **квадратичного отклонения** многочлена от таблично заданной функции.

$$\Phi = \sum_{j=0}^{N} \left[F_n(x_j) - y_j\right]^2.$$

Минимума можно добиться только за счет изменения коэффициентов многочлена $F_n(x)$. Необходимые условия экстремума имеют вид

$$\frac{\partial \Phi}{\partial a_k} = 2 \cdot \sum_{j=0}^{N} \left[\sum_{i=0}^{n} a_i x_j^i - y_j\right] x_j^k = 0, \quad k = 0, 1, \ldots, n.$$

Эту систему для удобства преобразуют к следующему виду:

$$\sum_{i=0}^{n} a_i \sum_{j=0}^{N} x_j^{k+i} = \sum_{j=0}^{N} y_j x_j^k, \quad k = 0, 1, \ldots, n.$$

Система в таком виде называется ***нормальной системой метода наименьших квадратов (МНК)***, представляет собой систему линейных алгебраических уравнений относительно коэффициентов a_i. Решив систему, построим многочлен $F_n(x)$, приближающий функцию $f(x)$ и минимизирующий квадратичное отклонение.

Необходимо отметить, что нормальная система МНК с увеличением степени n приближающего многочлена становится плохо обусловленной и решение её связано с большой потерей точности. Поэтому при использовании метода наименьших квадратов, как правило, используют *приближающий многочлен не выше третьей степени.*

2 Моя задача

Для таблично заданной функции

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
y_i	-1.8415	0.0	1.8415	2.9093	3.1411	3.2432

путем решения нормальной системы МНК найти приближающие многочлены а) 1-ой и б) 2-ой степени. Для каждого из приближающих многочленов вычислить сумму квадратов ошибок. Построить графики приближаемой функции и приближающих многочленов.

Решение:

Найдем приближающий многочлен первой степени $F_1(x) = a_0 + a_1 x$. Для нахождения неизвестных коэффициентов a_0, a_1 запишем нормальную систему МНК:

$$a_0(N+1) + a_1 \sum_{j=0}^{N} x_j = \sum_{j=0}^{N} y_j$$

$$a_0 \sum_{j=0}^{N} x_j + a_1 \sum_{j=0}^{N} x_j^2 = \sum_{j=0}^{N} y_j x_j$$

Мы имеем $N = 5,\ \ x_i, y_i,\ \ i = 0, \ldots, 5$ приведены в таблице. Тогда нормальная система НМК может быть записана следующим образом:

$$\begin{cases} 6a_0 + 9a_1 = 9.2936 \\ 9a_0 + 31a_1 = 31.8977 \end{cases}$$

Решение СЛАУ относительно a_i :

$$a_0 = 0.00973619, \quad a_1 = 1.02613.$$

Приближающий многочлен первой степени:

$$F_1(x) = 0.00973619 + 1.02613x$$

Найдем приближающий многочлен второй степени $F_1(x) = a_0 + a_1 x + a_2 x^2$. Для нахождения неизвестных коэффициентов $a_0, a_1, a_2 + a_2$ запишем нормальную систему МНК:

$$a_0(N+1) + a_1 \sum_{j=0}^{N} x_j + a_2 \sum_{j=0}^{N} x_j^2 = \sum_{j=0}^{N} y_j$$

$$a_0 \sum_{j=0}^{N} x_j + a_1 \sum_{j=0}^{N} x_j^2 + a_2 \sum_{j=0}^{N} x_j^3 = \sum_{j=0}^{N} y_j x_j$$

$$a_0 \sum_{j=0}^{N} x_j^2 + a_1 \sum_{j=0}^{N} x_j^3 + a_2 \sum_{j=0}^{N} x_j^4 = \sum_{j=0}^{N} y_j x_j^2$$

Мы имеем $N = 5$, x_i, y_i, $i = 0, \dots, 5$ приведены в таблице. Подставим числовые значения в СЛАУ:

$$\begin{cases} 6a_0 + 9a_1 + 31a_2 = 9.2936 \\ 9a_0 + 31a_1 + 99a_2 = 31.8977 \\ 31a_0 + 99a_1 + 355a_2 = 91.7983 \end{cases}$$

Решение СЛАУ относительно a_i :

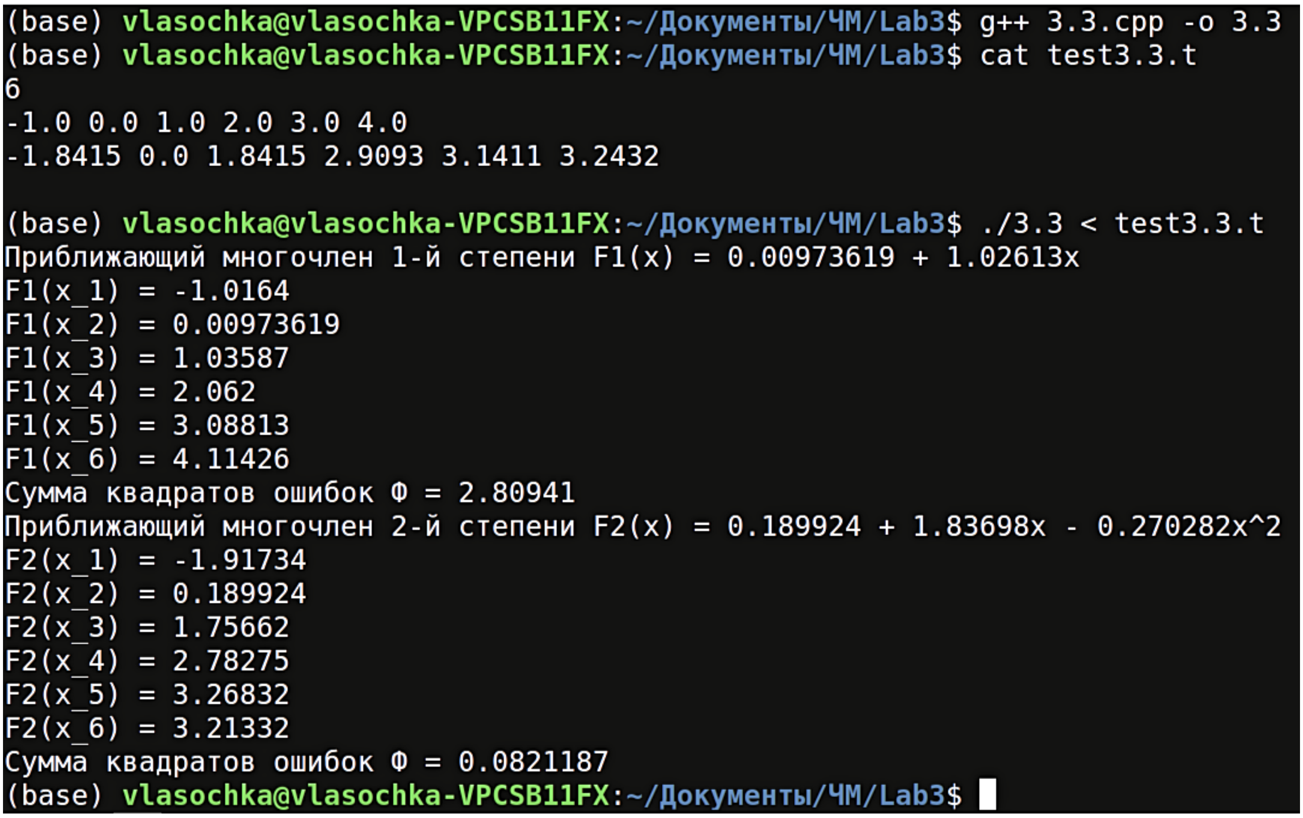
$$a_0 = 0.189924, \quad a_1 = 1.83698, \quad a_2 = -0.270282.$$

Приближающий многочлен второй степени:

$$F_2(x) = 0.189924 + 1.83698x - 0.270282x^2.$$

3 Протокол

Входные данные я храню в файле *test3.3.t*: первая строка — число интерполяционных узлов, вторая строка — их значения, третья строка — значения функции в этих узлах.
Выходные данные я вывожу на экран.
Скриншот консоли:



```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ g++ 3.3.cpp -o 3.3
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ cat test3.3.t
6
-1.0 0.0 1.0 2.0 3.0 4.0
-1.8415 0.0 1.8415 2.9093 3.1411 3.2432

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$ ./3.3 < test3.3.t
Приближающий многочлен 1-й степени F1(x) = 0.00973619 + 1.02613x
F1(x_1) = -1.0164
F1(x_2) = 0.00973619
F1(x_3) = 1.03587
F1(x_4) = 2.062
F1(x_5) = 3.08813
F1(x_6) = 4.11426
Сумма квадратов ошибок Φ = 2.80941
Приближающий многочлен 2-й степени F2(x) = 0.189924 + 1.83698x - 0.270282x^2
F2(x_1) = -1.91734
F2(x_2) = 0.189924
F2(x_3) = 1.75662
F2(x_4) = 2.78275
F2(x_5) = 3.26832
F2(x_6) = 3.21332
Сумма квадратов ошибок Φ = 0.0821187
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab3$
```

Приближающий многочлен 1-й степени

$$F_1(x) = 0.00973619 + 1.02613x$$

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
$F_1(x_i)$	-1.0164	0.00973619	1.03587	2.062	3.08813	4.11426

Сумма квадратов ошибок

$$\Phi = 2.80941$$

Приближающий многочлен 2-й степени

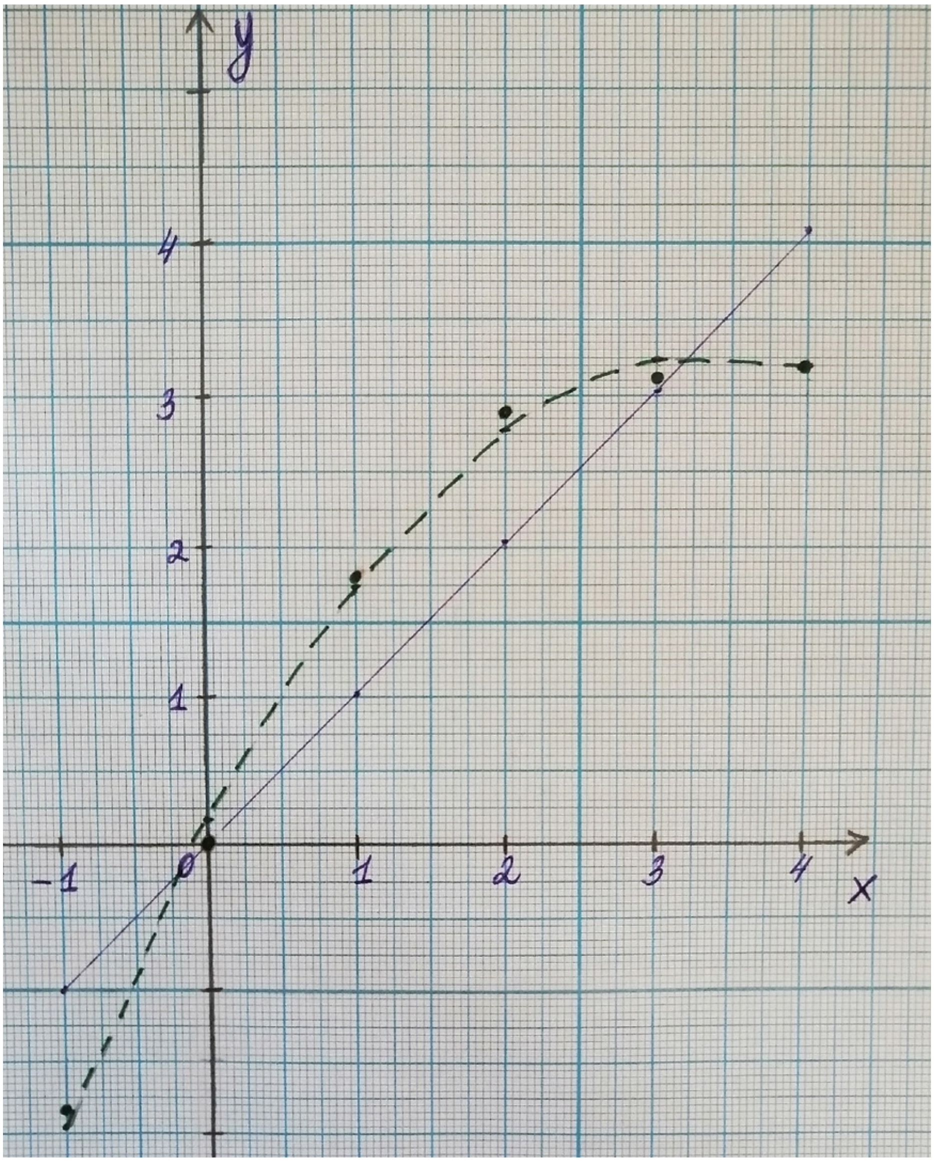
$$F_2(x) = 0.189924 + 1.83698x - 0.270282x^2$$

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
$F_2(x_i)$	-1.91734	0.189924	1.75662	2.78275	3.26832	3.21332

Сумма квадратов ошибок

$$\Phi = 0.0821187$$

На рисунке ниже точками обозначены табличные данные, сплошной линией — приближающий многочлен первой степени, пунктирной — приближающий многочлен второй степени.



4 Исходный код

Листинг 1: МНК

```
#include <vector>
#include <math.h>
#include <iostream>

void PrintF1(std::vector<double>& F1)
{
    std::cout << "Approximating polynomial of the 1st degree F1(x) = ";
    if(F1[0] != 0)
        std::cout << F1[0];

    if(F1[1] < 0)
        std::cout << " - " << -F1[1] << "x" << std::endl;
    else if(F1[1] > 0)
        std::cout << " + " << F1[1] << "x" << std::endl;
    else std::cout << std::endl;
}

void PrintF2(std::vector<double>& F2)
{
    std::cout << "Approximating polynomial of the 2nd degree F2(x) = ";
    if(F2[0] != 0)
        std::cout << F2[0];
    if(F2[1] < 0)
        std::cout << " - " << -F2[1] << "x";
    else if(F2[1] > 0)
        std::cout << " + " << F2[1] << "x";

    if(F2[2] < 0)
        std::cout << " - " << -F2[2] << "x^2" << std::endl;
    else if(F2[2] > 0)
        std::cout << " + " << F2[2] << "x^2" << std::endl;
    else std::cout << std::endl;
}

void Swap(std::vector<std::vector<double>>& Matrix, int i, int k)
{
    std::vector<double> H = Matrix[i];
    Matrix[i] = Matrix[k];
    Matrix[k] = H;
```

```
}

void Back(std::vector<std::vector<double>>& Matrix, std::vector<double>& x, int n)
{
    double sum = 0;
    int j = 0;
    for( int i = n -1; i >= 0; i--)
    {
        j = i + 1;
        while(j < n){
            sum += x[j] * Matrix[i][j];
            j++;
        }
        x[i] = ( Matrix[i][n] - sum) / Matrix[i][i];
        sum = 0;
    }
}

void Straight(std::vector<std::vector<double>>& Matrix, int n)
{
    int k = 0;
    double Kaf = 0;
    for( int i = 0; i < n - 1; i++)
    {
        k = i;
        while( Matrix[k][i] == 0 )
            k++;
        Swap(Matrix, i, k);
        for( int j = i + 1; j < n; j++)
        {
            Kaf = Matrix[j][i] / Matrix[i][i];
            for( int h = i; h <= n; h++)
                Matrix[j][h] -= Kaf * Matrix[i][h];
        }
    }
}

int main()
{
    int n;
    double mistake = 0;
    std::cin >> n;
    std::vector<double> x(n);
    std::vector<double> y(n);
```

```
    std::vector<double> F1(2, 0);
    std::vector<double> F2(3, 0);
    std::vector<std::vector<double>> A1(2, std::vector<double> (3, 0));
    std::vector<std::vector<double>> A2(3, std::vector<double> (4, 0));

    A1[0][0] = n;
    A2[0][0] = n;

    for(int i = 0; i < n; i++)
    {
        std::cin >> x[i];
        A1[0][1] += x[i];
        A1[1][1] += pow(x[i],2);
        A2[1][2] += pow(x[i], 3);
        A2[2][2] += pow(x[i], 4);
    }
    A1[1][0] = A1[0][1];
    A2[0][1] = A1[0][1];
    A2[1][0] = A2[0][1];
    A2[0][2] = A1[1][1];
    A2[1][1] = A2[0][2];
    A2[2][0] = A2[0][2];
    A2[2][1] = A2[1][2];

    for(int i = 0; i < n; i++)
    {
        std::cin >> y[i];
        A1[0][2] += y[i];
    }
    A2[0][3] = A1[0][2];
    for(int i = 0; i < n; i++)
    {
        A1[1][2] += x[i]*y[i];
        A2[2][3] += y[i]*pow(x[i], 2);
    }
    A2[1][3] = A1[1][2];
    Straight(A1, 2);
    Back(A1, F1, 2);
    PrintF1(F1);

    for(int i = 0; i < n; i++)
        std::cout << "F1(x_" << i + 1 << ") = " << F1[0] + F1[1]*x[i] << std::endl;

    for(int i = 0; i < n; i++)
```

```
            mistake += pow(F1[0] + F1[1]*x[i] - y[i], 2);
      std::cout << "Sum of error squares  = " << mistake << std::endl;
      mistake = 0;

      Straight(A2, 3);
      Back(A2, F2, 3);
      PrintF2(F2);

      for(int i = 0; i < n; i++)
            std::cout << "F2(x_" << i + 1 << ") = " << F2[0] + F2[1]*x[i] + F2[2]*pow(x[i], 2)
                  << std::endl;

      for(int i = 0; i < n; i++)
            mistake += pow(F2[0] + F2[1]*x[i] + F2[2]*pow(x[i], 2) - y[i], 2);
      std::cout << "Sum of error squares  = " << mistake << std::endl;

      return 0;
}
```

5 Выводы

Выполнив третье задание третьей лабораторной работы, я изучила метод наименьших квадратов, который позволяет интерполировать таблично заданную функцию, значения которой определены с некоторой погрешностью, так же известен вид функции — многочлен некоторой степени n. Неизвестные коэффициенты функции ищутся из условия минимума квадратичного отклонения многочлена от таблично заданной функции. Таким образом, для определения коэффициентов приближающего многочлена решается нормальная система МНК — система линейных алгебраических уравнений. Система с увеличением степени n приближающего многочлена становится плохо обусловленной, и решение её связано с большой потерей точности. Поэтому при использовании метода наименьших квадратов, как правило, используют приближающий многочлен не выше третьей степени.