**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №3

**Метод наименьших квадратов**

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Ассистентка кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2022

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 2-3

Листинг программы ---------------------------------------------------------------- 3-6

Результаты --------------------------------------------------------------------------- 7

Выводы ------------------------------------------------------------------------------- 7

**Постановка задачи**

На отрезке [a, b] задана таблица значений функции f(x) с шагом h = 0,1. По заданной таблице значений найти наилучшие среднеквадратичные приближения при n = 4, 6. Найти . В содержание отчета должна быть включена следующая информация:

• Метод наименьших квадратов.

• Графики аппроксимирующих функций и график заданной функции (по множеству точек).

• .

• Листинг программы с комментариями.

Вариант 4.

Функция . Отрезок

**Краткие теоретические сведения**

*Метод наименьших квадратов:*

Даны

Выбираем конечную линейно независимую систему базисных функций .

– наилучшее среднеквадратичное приближение.

Находим коэффициенты из системы:

Где .

Если в качестве системы базисных функций выбрать

Где

**Листинг программы**

Файл Function.java:

import java.util.\*;  
  
public class Function {  
 private static final int *N* = 40;  
 private static final int *n1* = 4;  
 private static final int *n2* = 6;  
 private double[] c1;  
 private double[] c2;  
 private final double[] xk;  
 private final double[] fk;  
  
 public Function() {  
 //Вектор узлов  
 xk = new double[*N* + 1];  
 double h = 0.1;  
 xk[0] = -2.;  
 for(int i = 1; i < *N* + 1; i++) {  
 xk[i] = xk[i - 1] + h;  
 }  
  
 //Вектор значений  
 fk = new double[*N* + 1];  
 for(int i = 0; i < *N* + 1; i++) {  
 fk[i] = f(xk[i]);  
 }  
 }  
  
 public void formQs() {  
 //Находим коэффициенты для Q(4)  
 c1 = findC(*n1*);  
 //Находим коэффициенты для Q(6)  
 c2 = findC(*n2*);  
 }  
  
 public void outQs() {  
 Formatter fmt = new Formatter();  
 fmt.format("Функция Q4:\n");  
 fmt.format("Q4(x) = (%.16f)\n", c1[0]);  
 fmt.format("+ (%.16f) \* x\n", c1[1]);  
 fmt.format("+ (%.16f) \* x^2\n", c1[2]);  
 fmt.format("+ (%.16f) \* x^3\n", c1[3]);  
 fmt.format("+ (%.16f) \* x^4.\n", c1[4]);  
 fmt.format("Приращение d^2(f) = %.16f.\n", getD1());  
 fmt.format("Функция Q6:\n");  
 fmt.format("Q6(x) = (%.16f)\n", c2[0]);  
 fmt.format("+ (%.16f) \* x\n", c2[1]);  
 fmt.format("+ (%.16f) \* x^2\n", c2[2]);  
 fmt.format("+ (%.16f) \* x^3\n", c2[3]);  
 fmt.format("+ (%.16f) \* x^4\n", c2[4]);  
 fmt.format("+ (%.16f) \* x^5\n", c2[5]);  
 fmt.format("+ (%.16f) \* x^6.\n", c2[6]);  
 fmt.format("Приращение d^2(f) = %.16f.\n", getD2());  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
  
 private double[] findC(int n) {  
 //Формируем матрицу из s[i]  
 double[][] s = new double[n + 1][n + 1];  
 s[0][0] = *N* + 1;  
 double si;  
 for(int i = 1; i < n + 1; i++) {  
 si = 0.;  
 for(int j = 0; j < *N* + 1; j++) {  
 si += Math.*pow*(xk[j], i);  
 }  
 s[0][i] = si;  
 for(int j1 = 1, j2 = i - 1; j1 < n + 1 && j2 >= 0; j1++, j2--) {  
 s[j1][j2] = si;  
 }  
 }  
 for(int i = 1; i < n + 1; i++) {  
 si = 0.;  
 for(int j = 0; j < *N* + 1; j++) {  
 si += Math.*pow*(xk[j], n + i);  
 }  
 s[i][n] = si;  
 for(int j1 = i + 1, j2 = n - 1; j1 < n + 1 && j2 >= 0; j1++, j2--) {  
 s[j1][j2] = si;  
 }  
 }  
  
 //Формируем вектор из m[i]  
 double[] m = new double[n + 1];  
 double mi;  
 for(int i = 0; i < n + 1; i++) {  
 mi = 0.;  
 for(int j = 0; j < *N* + 1; j++) {  
 mi += fk[j] \* Math.*pow*(xk[j], i);  
 }  
 m[i] = mi;  
 }  
  
 //Решаем систему, находим коэффициенты  
 return gauss(s, m);  
 }  
  
 private double getD1() {  
 //Подсчёт приращения для Q(4)  
 double d = 0;  
 for(int i = 0; i < *N* + 1; i++) {  
 d += Math.*pow*(fk[i] - q1(xk[i]), 2);  
 }  
 return d;  
 }  
  
 private double getD2() {  
 //Подсчёт приращения для Q(6)  
 double d = 0;  
 for(int i = 0; i < *N* + 1; i++) {  
 d += Math.*pow*(fk[i] - q2(xk[i]), 2);  
 }  
 return d;  
 }  
  
 private double[] gauss(double[][] a, double[] f) throws NumberFormatException {  
 //Метод Гаусса  
 double max;  
 int kMax;  
 double[] temp;  
 double temp2;  
  
 //Прямой ход  
 for (int i = 0; i < a.length; i++) {  
 //Поиск максимального по модулю элемента (выбор главного элемента по столбцу)  
 kMax = i;  
 max = 0;  
 for (int k = i; k < a.length; k++) {  
 if (Math.*abs*(a[k][i]) > Math.*abs*(max)) {  
 max = a[k][i];  
 kMax = k;  
 }  
 }  
 //Проверка на невозможность применения  
 if (max == Double.*MIN\_VALUE*) {  
 System.*out*.println("Ошибка. Деление на ноль.");  
 throw new NumberFormatException();  
 }  
 //Перестановка строк  
 if (max != a[i][i]) {  
 temp = a[i];  
 a[i] = a[kMax];  
 a[kMax] = temp;  
 temp2 = f[i];  
 f[i] = f[kMax];  
 f[kMax] = temp2;  
 }  
  
 //Деление на главный элемент  
 for (int j = i + 1; j < a.length; j++) {  
 a[i][j] /= a[i][i];  
 }  
 f[i] /= a[i][i];  
  
 //Обнуление столбца  
 for (int j = i + 1; j < a.length; j++) {  
 for (int k = i + 1; k < a.length; k++) {  
 a[j][k] -= a[j][i] \* a[i][k];  
 }  
 f[j] -= a[j][i] \* f[i];  
 }  
 }  
  
 //Обратный ход  
 double[] x = new double[a.length];  
 x[a.length - 1] = f[a.length - 1];  
 for (int i = a.length - 2; i >= 0; i--) {  
 x[i] = f[i];  
 for (int j = a.length - 1; j > i; j--) {  
 x[i] -= a[i][j] \* x[j];  
 }  
 }  
  
 return x;  
 }  
  
 private double f(double x) {  
 //Функция f(x)  
 return Math.*sin*(x) \* Math.*cos*(x);  
 }  
  
 private double q1(double x) {  
 //Функция Q4(x)  
 double res = 0;  
 for(int i = 0; i < *n1* + 1; i++) {  
 res += c1[i] \* Math.*pow*(x, i);  
 }  
 return res;  
 }  
  
 private double q2(double x) {  
 //Функция Q6(x)  
 double res = 0;  
 for(int i = 0; i < *n2* + 1; i++) {  
 res += c2[i] \* Math.*pow*(x, i);  
 }  
 return res;  
 }  
}

Файл Main.java:

public class Main {  
  
 public static void main(String[] args) {  
 Function f = new Function();  
 f.formQs();  
 f.outQs();  
 }  
}

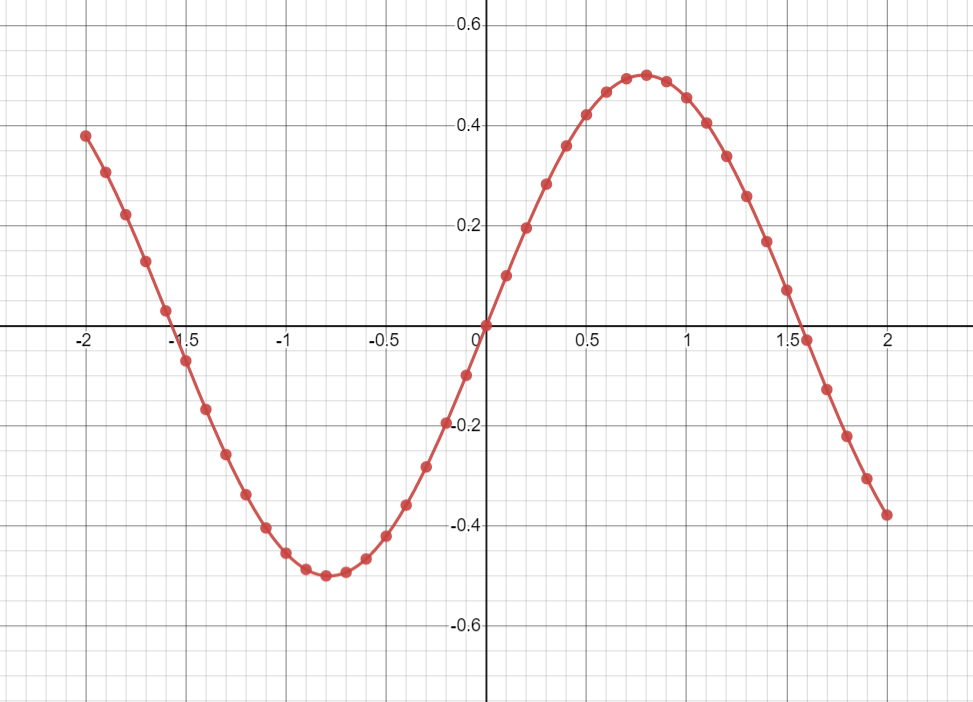
}

**Результаты**

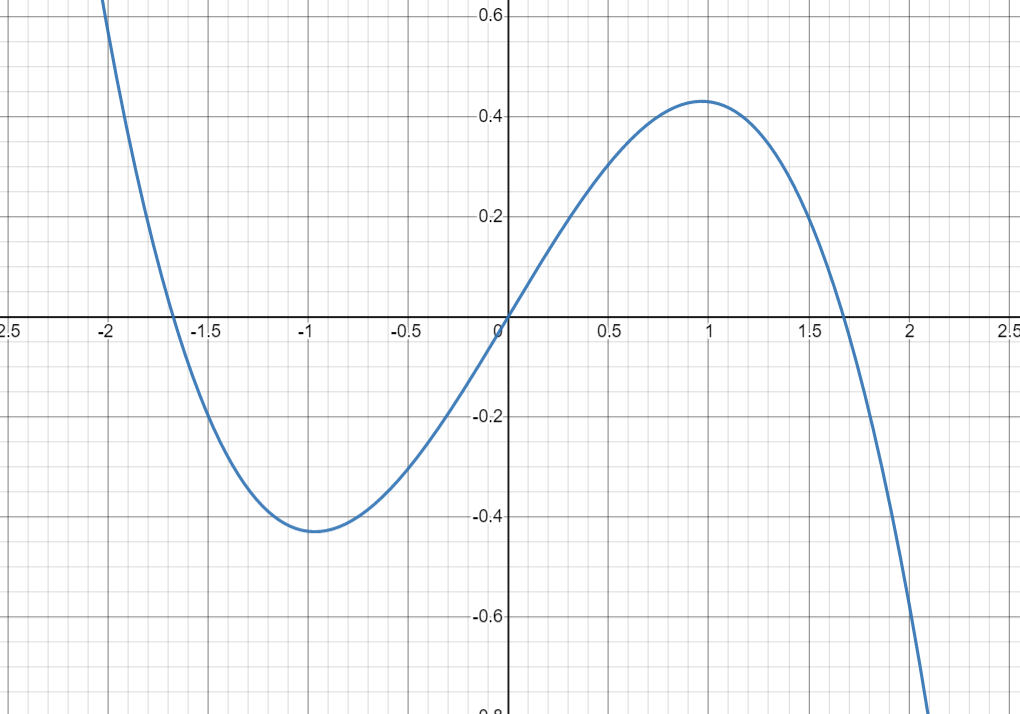
****

Графики:

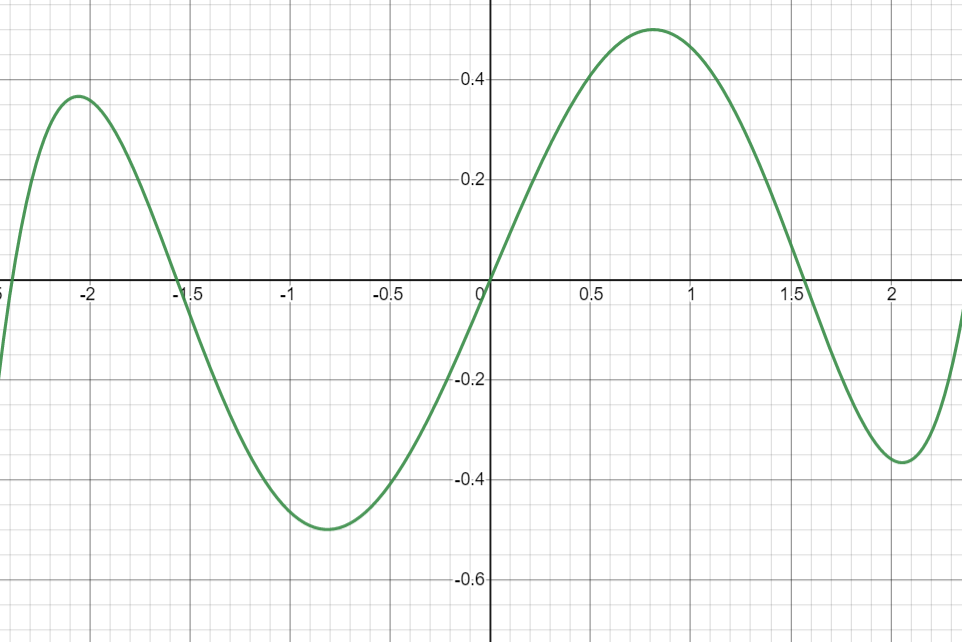
f(x):

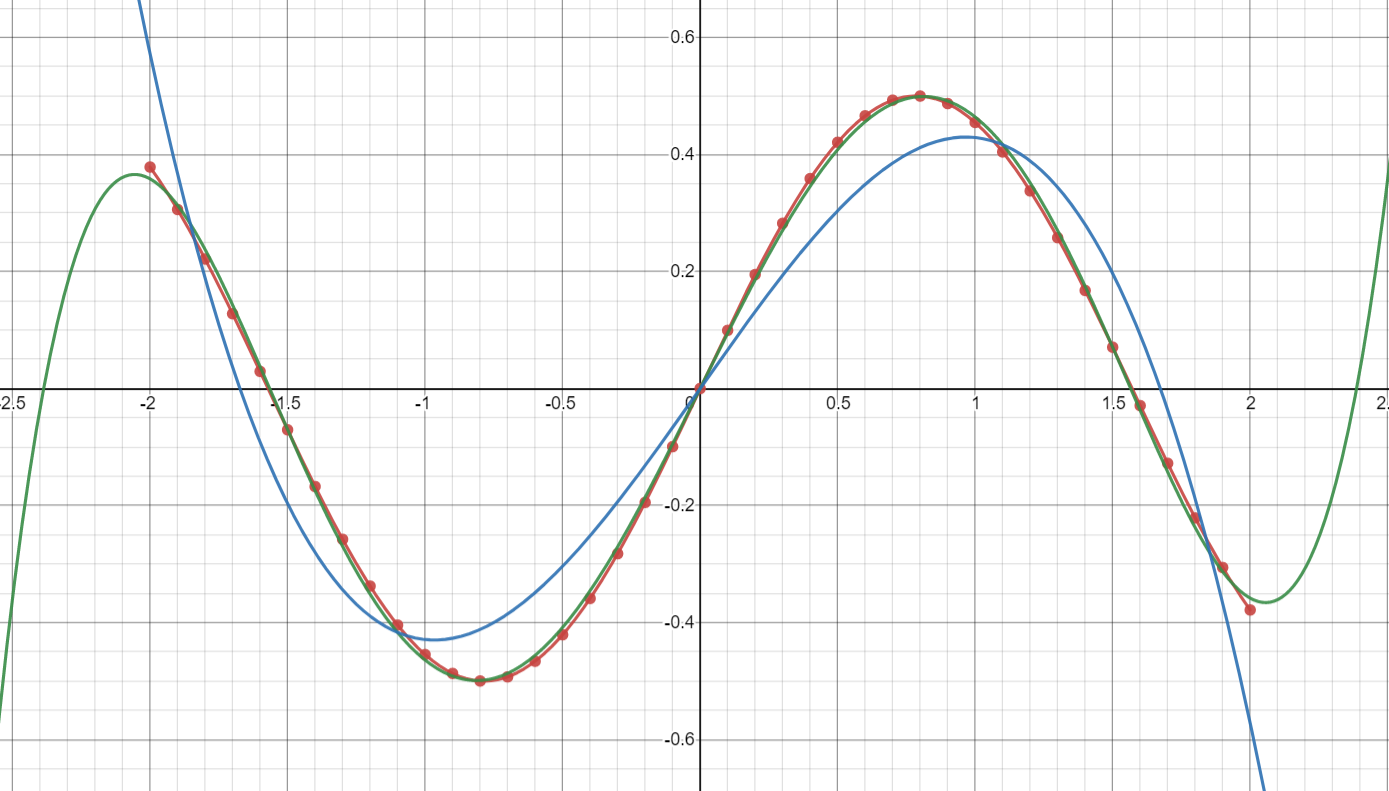


:



:





Красный – , синий - , зелёный -

**Выводы**

1) При больших n приближение по методу наименьших квадратов становится точнее.

2) Полученная система является положительно определённой, поэтому её оптимально решать методом квадратного корня.