**Лабораторная работа № 3**

**«Метод наименьших квадратов»**

**Чеботаревский Никита, 2 курс, 7 группа**

**Постановка задачи**

На отрезке [a, b] задана таблица значений функций f(x) с шагом h = 0.1. По заданной таблице значений найти наилучшие среднеквадратичные приближения при n = 2, 5. Найти

В содержании отчёта привести следующую информацию:

1. Метод наименьших квадратов
2. Графики аппроксимирующих функций и график заданной функции (по множеству точек).
3. Листинг программы с комментариями

Применить метод наименьших квадратов для функции:



на отрезке [-2, 2].

**Теоретические сведения**

**Метод наименьших квадратов:**

Пусть дан отрезок [a, b].

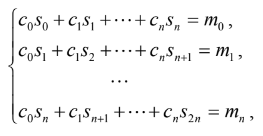
, f() = , i = , .

(f, g) =

– многочлен наилучшего среднеквадратичного приближения.

- величина наилучшего среднеквадратичного приближения.

Коэффициенты многочлена наилучшего среднеквадратичного приближения могут быть найдены как решение системы, которая имеет вид:

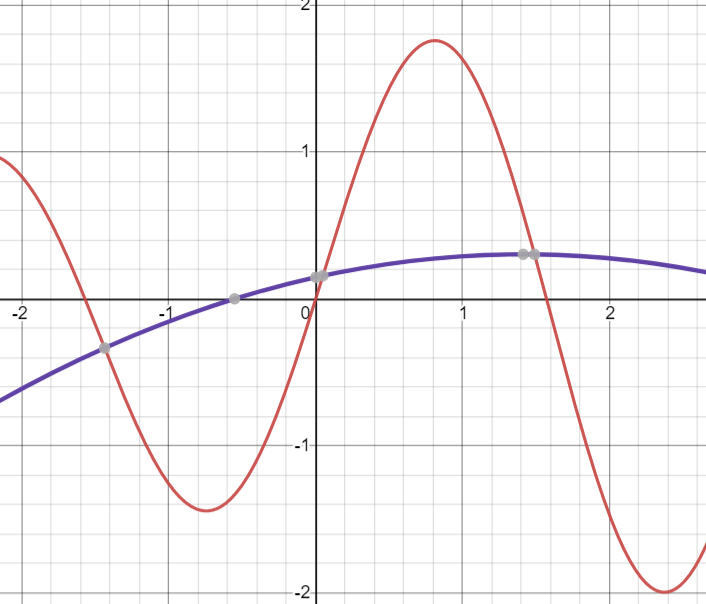


которая имеет единственное решение.

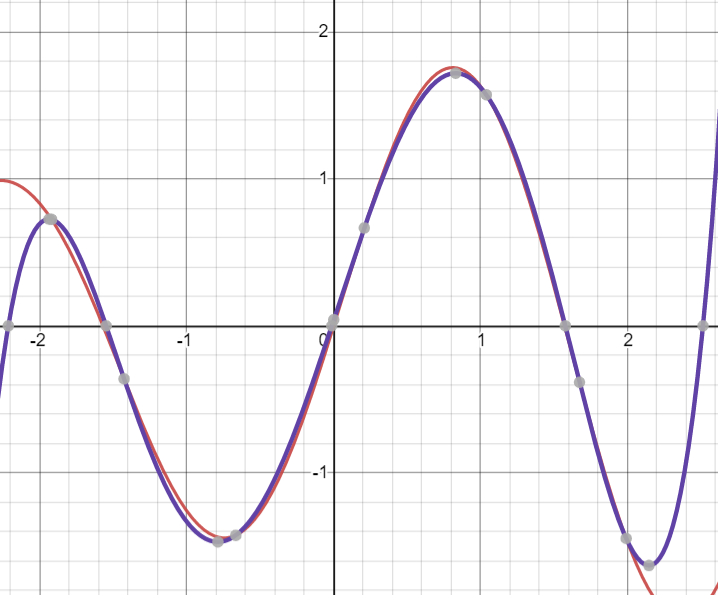
В данной работе будем считать, что

**Графики аппроксимирующих функций и заданной функции**

При n = 2: (величина наилучшего среднеквадратичного приближения равна: 43.21945289025507)



При n = 5: (величина наилучшего среднеквадратичного приближения равна: 0.1037476968463813)



**Листинг программы**

*import* numpy *as* np  
*from* math *import* \*  
  
  
*def* f(argument):  
 *return* sin(2 \* argument) \* log(argument + 5, e)  
  
  
*def* count\_s(array\_x, size):  
 s = [0] \* (2 \* size + 1)  
 *for* j *in* range(2 \* size + 1):  
 num = 0  
 *for* k *in* range(len(array\_x)):  
 num += array\_x[k] \*\* j  
 s[j] = num  
 *return* s  
  
  
*def* count\_m(arr\_x, arr\_f, number):  
 m = [0] \* (number + 1)  
 *for* j *in* range(number + 1):  
 num = 0  
 *for* k *in* range(len(arr\_x)):  
 num += arr\_f[k] \* arr\_x[k] \*\* j  
 m[j] = num  
  
 *return* m  
  
  
*def* create\_matrix(array\_s, size):  
 matrix = []  
 *for* index *in* range(size + 1):  
 temp = []  
 *for* ind *in* range(size + 1):  
 temp.append(array\_s[ind + index])  
 matrix.append(temp)  
 *return* matrix  
  
  
*def* count\_q(array\_c, arg):  
 q = 0  
 *for* index *in* range(len(array\_c)):  
 q += array\_c[index] \* arg \*\* index  
 *return* q  
  
  
*def* count\_delta(arr\_x, arr\_c):  
 delta = 0  
 *for* index *in* range(len(arr\_x)):  
 delta += (f(arr\_x[index]) - count\_q(arr\_c, arr\_x[index])) \*\* 2  
 *return* delta  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 x = []  
 start\_number = -2  
 *while* start\_number < 2:  
 x.append(start\_number)  
 start\_number += 0.1  
 x.append(2)  
  
 fx = []  
 *for* i *in* range(len(x)):  
 fx.append(f(x[i]))  
 b2 = np.array(count\_m(x, fx, 2))  
 A2 = np.array(create\_matrix(count\_s(x, 2), 2))  
 c2 = np.linalg.inv(A2).dot(b2)  
 print("Coefficients si with n = 2: ")  
 print(count\_s(x, 2))  
 print("Coefficients mi with n = 2: ")  
 print(count\_m(x, fx, 2))  
 print(f"Coefficients ci with n = 2: {c2}")  
 print(f"Delta with n = 2: {count\_delta(x, c2)}")  
 print()  
  
 b5 = np.array(count\_m(x, fx, 5))  
 A5 = np.array(create\_matrix(count\_s(x, 5), 5))  
 c5 = np.linalg.inv(A5).dot(b5)  
 print("Coefficients si with n = 5: ")  
 print(count\_s(x, 5)[0: 5])  
 print(count\_s(x, 5)[5:])  
 print("Coefficients mi with n = 5: ")  
 print(count\_m(x, fx, 5))  
 print(f"Coefficients ci with n = 5: {c5}")  
 print(f"Delta with n = 5: {count\_delta(x, c5)}")

**Результаты программы**

