МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Национальный исследовательский ядерный университет

«МИФИ» (НИЯУ МИФИ)

Институт финансовых технологий и экономической безопасности
Кафедра финансового мониторинга

Лабораторная работа №3:

По курсу “Численные методы”

Работу выполнил: студент группы С21-762: Ле К.Х.

Проверил: Саманчук В.Н.

Москва 2023

## Постановка задачи

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| У | 5 | 6 | 8 | 10 | 12 | 13 | 12 | 10 | 8 | 10 | 8 | 11 | 7 | 9 | 11 | 10 | 9 | 12 | 11 | 6 |

Необходимо интерполировать таблично заданную функцию, используя полином Лагранжа седьмого порядка, с шагом 0,25, то есть для каждого частичного отрезка дополнительно получить 3 интерполированных отсчета – например, для отрезка [ 1, 2 ] это точки (по оси x) 1,25 1,5 1,75; для отрезка [ 2, 3 ] это точки 2,25 2,5 2,75 и так далее, а затем построить график получившейся функции.

## Методика решения

Для решения поставленной задачи была написана программа на языке Python, в которой реализована интерполяция функции с помощью полинома Лагранжам седьмого порядка.

## Теоретическая справка

*Интерполяция* – способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

Пусть в ходе эксперимента при изменении входной величины $x$ ($x_0$, $x_1$, $x_2$,..., $x_n$) получены значения функции $y=f(x)$ ($y_0$, $y_1$, $y_2$....$y_n$) (табл. 1).

Таблица 1

Вид таблицы экспериментальных данных

| | | | | | |
|---|---|---|---|---|---|
| $x_0$ | $x_1$ | $x_2$ | ... | $x_{n\text{-}1}$ | $x_n$ |
| $y_0$ | $y_1$ | $y_2$ | ... | $y_{n\text{-}1}$ | $y_n$ |

Интерполяцию функций применяют в случае, когда требуется найти значение функции $y(x)$ при значении аргумента $x_i$, принадлежащего интервалу [$x_0$, ..., $x_n$], но не совпадающего по значению ни с одним значением, приведенным в таблице 1.

Данная задача, а именно интерполяция функций, часто встречается при ограниченности возможностей при проведении эксперимента. В частности из-за дороговизны и трудоемкости проведения эксперимента размер выборки ($x_0$, $x_1$, $x_2$,..., $x_n$) может быть достаточно мал.

При этом во многих случаях аналитическое выражение функции $y(x)$ не известно и получить его по таблице ее значений (табл. 1) в большинстве случаев невозможно. Поэтому вместо нее строят другую функцию, которая легко вычисляется и имеет ту же таблицу значений (совпадает с ней в точках $x_0$, $x_1$, $x_2$,..., $x_n$), что и $f(x)$, т. е.

$$P_n(x_0)=f(x_0)=y_0;$$

$$\dots \qquad (1)$$

$$P_n(x_i)=f(x_i)=y_i;$$

где $i = 0, 1, 2, \dots, n$.

Нахождение приближенной функции называется интерполяцией, а точки $x_0, x_1, x_2, \dots, x_n$ – узлами интерполяции.

Интерполирующую функцию ищут в виде полинома $n$ степени.

Для каждого набора точек имеется только один интерполяционный многочлен, степени не больше $n$. Однозначно определенный многочлен может быть представлен в различных видах.

Графически задача интерполирования заключается в том, чтобы построить такую интерполирующую функцию, которая бы проходила через все узлы интерполирования (рис. 1).

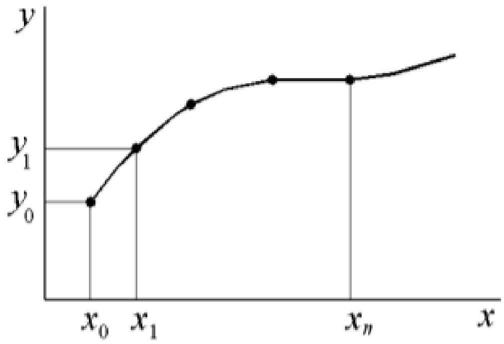



Рис. 1. Вид интерполирующей функции

Числитель и знаменатель не должны включать в себя значения $x=x_i$, так как результат будет равен нулю.

Интерполяционный полином Лагранжа обычно применяется в теоретических исследованиях (при доказательстве теорем, аналитическом решении задач и т. п.).

Интерполяционный многочлен Лагранжа имеет вид:

$$y(x)=\sum_{j=1}^{n+1} L_j(x)\cdot y_j, \text{ где } L_j(x)=\begin{cases} 1, & \text{если } x = x_j \\ 0, & \text{если } x = x_i, \quad i \neq j \end{cases}$$

$$L_j(x)=\frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{j-1})\cdot(x-x_{j+1})\cdot\ldots\cdot(x-x_{n+1})}{(x_j-x_1)\cdot(x_j-x_2)\cdot\ldots\cdot(x_j-x_{j-1})\cdot(x_j-x_{j+1})\cdot\ldots\cdot(x_j-x_{n+1})} = \frac{\prod_{i=1;\, i\neq j}^{n+1}(x-x_i)}{\prod_{i=1;\, i\neq j}^{n+1}(x_j-x_i)}$$

Числитель и знаменатель не должны включать в себя $x = x_i$, так как результат будет равен нулю.

В нашем случае n=7, $y(x)=\sum_{i=1}^{8} L_j(x)\cdot y_j$.

**Практическая реализация.** Многочлен Лагранжа рассчитывается по n+1 (8) точкам.

1. Начало таблицы.
   Получаем значения функции в интервалах $(x_1, x_2)$, $(x_2, x_3)$, $(x_3, x_4)$, $(x_4, x_5)$.
2. Середина таблицы. Путем передвижения шаблона получаем промежуточные значения функции в интервалах $(x_5, x_6)$, …, $(x_{15}, x_{16})$.
3. Конец таблицы. Получаем значения функции в интервалах $(x_{16}, x_{17})$, $(x_{17}, x_{18})$, $(x_{18}, x_{19})$, $(x_{19}, x_{20})$.

## Решение задачи

```
import numpy as np
import matplotlib.pyplot as plt

A = np.zeros((2, 77))
for i in range(77):
    A[0, i] = 1+0.25*i
A[1, 0] = 5; A[1, 4] = 6; A[1, 8] = 8; A[1, 12] = 10; A[1, 16] = 12; A[1, 20]
= 13; A[1, 24] = 12; A[1, 28] = 10
A[1, 32] = 8; A[1, 36] = 10; A[1, 40] = 8; A[1, 44] = 11; A[1, 48] = 7; A[1,
52] = 9; A[1, 56] = 11
A[1, 60] = 10; A[1, 64] = 9; A[1, 68] = 12; A[1, 72] = 11; A[1, 76] = 6

def L(x, i, j):
    l = float(1)
    for k in range(j, j+8):
        if (k != i):
            l *= (x-A[0, 4*k])/(A[0, 4*i]-A[0, 4*k])
    return l
def f(x, j):
    f = 0
    for i in range(j, j+8):
        f += L(x, i, j)*A[1, 4*i]
    return f

for i in range(16):
    A[1, i] = f(A[0, i], 0)

for i in range(1, 12):
    A[1, (i+3)*4+1] = f(A[0, (i+3)*4+1], i)
    A[1, (i+3)*4+2] = f(A[0, (i+3)*4+2], i)
    A[1, (i+3)*4+3] = f(A[0, (i+3)*4+3], i)

for i in range(60, 76):
    A[1, i] = f(A[0, i], 12)

for i in range(0, 77):
    print('f(', A[0, i], ')=', A[1, i])

plt.plot(A[0], A[1])
plt.xlabel("x")
plt.ylabel("f(x)")
for i in range(0, 20):
    plt.plot(A[0, i*4], A[1, i*4], 'ro')
plt.show()
```

## Результат работы

x= 1.0; y= 5.0

x= 1.25; y= 4.9039230346679705

x= 1.5; y= 5.125

x= 1.75; y= 5.521125793457031

x= 2.0; y= 6.0

x= 2.25; y= 6.506797790527344

x= 2.5; y= 7.013671875

x= 2.75; y= 7.5109634399414045

x= 3.0; y= 8.0

x= 3.25; y= 8.487358093261719

x= 3.5; y= 8.98046875

x= 3.75; y= 9.48444366455078

x= 4.0; y= 10.0

x= 4.25; y= 10.522361755371094

x= 4.5; y= 11.041015625

x= 4.75; y= 11.540199279785154

x= 5.0; y= 12.0

x= 5.25; y= 12.402851104736328

x= 5.5; y= 12.71826171875

x= 5.75; y= 12.923152923583983

x= 6.0; y= 13.0

x= 6.25; y= 12.916606903076172

x= 6.5; y= 12.70849609375

x= 6.75; y= 12.395298004150389

x= 7.0; y= 12.0

x= 7.25; y= 11.597774505615234

x= 7.5; y= 11.12939453125

x= 7.75; y= 10.593067169189451

x= 8.0; y= 10.0

x= 8.25; y= 9.241645812988281

x= 8.5; y= 8.5810546875

x= 8.75; y= 8.137901306152344

x= 9.0; y= 8.0

x= 9.25; y= 8.482006072998047

x= 9.5; y= 9.11474609375

x= 9.75; y= 9.684513092041014

x= 10.0; y= 10.0

x= 10.25; y= 9.51077651977539

x= 10.5; y= 8.84912109375

x= 10.75; y= 8.267192840576172

x= 11.0; y= 8.0

x= 11.25; y= 8.733543395996094

x= 11.5; y= 9.6865234375

x= 11.75; y= 10.54346466064453

x= 12.0; y= 11.0

x= 12.25; y= 10.299240112304688

x= 12.5; y= 9.19140625

x= 12.75; y= 7.9816436767578125

x= 13.0; y= 7.0

x= 13.25; y= 6.922824859619141

x= 13.5; y= 7.32275390625

x= 13.75; y= 8.075572967529297

x= 14.0; y= 9.0

x= 14.25; y= 9.740497589111328

x= 14.5; y= 10.35888671875

x= 14.75; y= 10.789730072021483

x= 15.0; y= 11.0

x= 15.25; y= 11.00634765625

x= 15.5; y= 10.80859375

x= 15.75; y= 10.450927734374998

x= 16.0; y= 10.0

x= 16.25; y= 9.471965789794922

x= 16.5; y= 9.06005859375

x= 16.75; y= 8.87692642211914

x= 17.0; y= 9.0

x= 17.25; y= 9.452816009521483

x= 17.5; y= 10.19189453125

x= 17.75; y= 11.101673126220703

x= 18.0; y= 12.0

x= 18.25; y= 12.656688690185547

x= 18.5; y= 12.82763671875

x= 18.75; y= 12.307010650634764

x= 19.0; y= 11.0

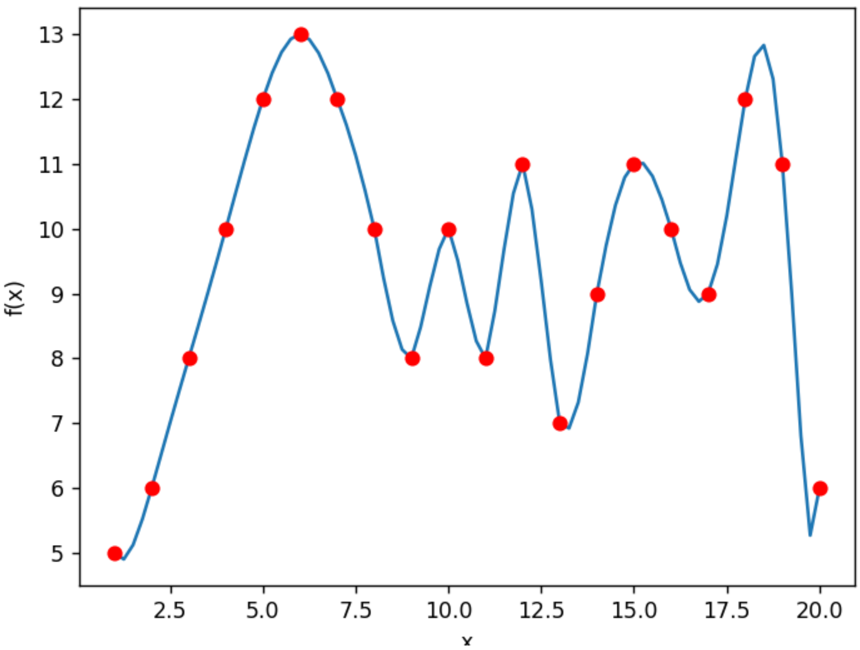
x= 19.25; y= 9.01864242553711

x= 19.5; y= 6.8032226562499964

x= 19.75; y= 5.271747589111327

x= 20.0; y= 6.

Красные точки – табличные значения, синяя линия - интерполирующая функция




## Заключение

В задании требовалось вычислить интерполированные значения таблично заданной функции с шагом 0,25, а затем построить график получившейся функции. Для решения задачи была написана программа на языке программирования Python.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Национальный исследовательский ядерный университет «МИФИ» (НИЯУ МИФИ)

Институт Финансовых Технологий и Экономической Безопасности Кафедра Финансового мониторинга

# Лабораторная работа №4

# «Решение задачи аппроксимации»

Выполнил студент группы С21-762: Ле К.Х.

Проверил Саманчук В.Н.

Москва, 2023

## Задание

Аппроксимировать таблично заданную функцию по методу наименьших квадратов, используя полиномы Лежандра по пятый порядок включительно.

Таблица 1

| X | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| У | 5 | 6 | 8 | 10 | 12 | 13 | 12 | 10 | 8 | 10 |

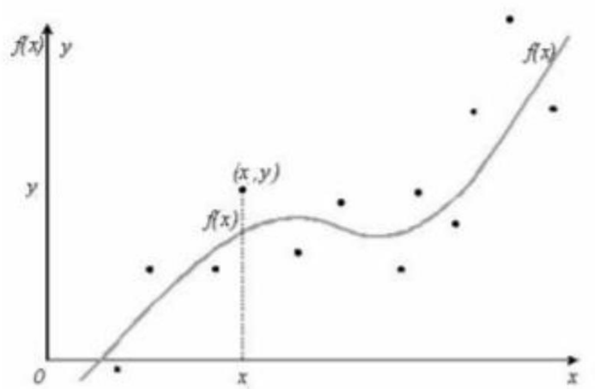
продолжение таблицы 1

| X | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| У | 8 | 11 | 7 | 9 | 11 | 10 | 9 | 12 | 11 | 6 |

После решения задачи аппроксимации нужно построить график аппроксимирующей кривой и, дополнительно, подсчитать сумму квадратов отклонений между значениями функции и аппроксимирующей кривой в узловых точках.

## Описание метода

При аппроксимации желательно получить относительно простую функциональную зависимость (например, многочлен), которая позволила бы «сгладить» экспериментальные погрешности, вычислять значения функции в точках, не содержащихся в исходной таблице.
Эта функциональная зависимость должна с достаточной точностью соответствовать исходной табличной зависимости. В качестве критерия точности чаще всего используют критерий *наименьших квадратов*, т.е. определяют такую функциональную зависимость $f(x)$, при которой $R = \sum_{i=0}^{n}\left(y_i - f(x_i)\right)^2$ обращается в минимум.




Рассмотрим в качестве функциональной зависимости многочлен.
$P_m(x) = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m$, тогда

$$R + \sum_{i=0}^{n}\left(y_i - P_m(x_i)\right)^2.$$

Условия минимума - нулевые частные производные по всем переменным $a_0, a_1, a_2, \dots, a_m$. Т.е.

$$\frac{\partial R}{\partial a_k} = -\sum_{i=0}^{n} 2(y_i - a_0 - a_1x_i - \cdots - a_mx_i)x_i^k = 0.$$

или

$$\sum_{i=0}^{n}(y_i - a_0 - a_1x_i - \cdots - a_mx_i)x_i^k = 0, \qquad k = 0,1,2,\dots,m.$$

Собираем коэффициенты при неизвестных $a_0, a_1, a_2, \dots, a_m$ получаем систему уравнений:

$$a_0\sum_{i=0}^{n}x_i^k+a_1\sum_{i=0}^{n}x_i^{k+1}+a_2\sum_{i=0}^{n}x_i^{k+2}+\cdots+a_m\sum_{i=0}^{n}x_i^{k+m}=\sum_{i=0}^{n}x_i^k y_i\,,\qquad k=0,1,2,\dots,m.$$

Можно ввести обозначения:
$c_k=\sum_{i=0}^{n}x_i^k$, $b_k=\sum_{i=0}^{n}x_i^k y_i$ и переписать систему в развернутом виде.

$$\begin{cases} c_0a_0+c_1a_1+c_2a_2\cdots+c_ma_m=b_0\\ c_1a_0+c_2a_1+c_3a_2+\cdots+c_{m+1}a_m=b_1\\ \cdots\\ c_ma_0+c_{m+1}\,a_1+c_{m+2}\,a_2+\cdots+c_{2m}a_m=b_m\end{cases}$$

Матрица данной системы называется матрицей Грамма. Решая эту систему линейных уравнений, получаем коэффициенты $a_0,a_1,a_2,\dots,a_m$, которые являются искомыми параметрами эмпирической формулы.
Полиномы Лежандра, представляемые в виде разложения по степеням *x,* можно преобразовать к весьма компактному виду, известному под названием *формулы Родрига:*

$$P_l(x)=\frac{1}{2^l l!}\frac{d^l}{dx^l}(x^2-1)^l\,.$$

Применим метод наименьших квадратов

$$y(x)\to\Phi_m(x)=\sum_{i=0}^{m}c_i\varphi_i(x)$$

Вид системы линейных уравнений

$$\begin{cases}\sum_{i=1}^{n}\varphi_0(x_i)\varphi_0(x_i)c_0+\sum_{i=1}^{n}\varphi_0(x_i)\varphi_1(x_i)c_1+\cdots+\sum_{i=1}^{n}\varphi_0(x_i)\varphi_m(x_i)c_m=\sum_{i=1}^{n}y_i\varphi_0(x_i)\\ \sum_{i=1}^{n}\varphi_1(x_i)\varphi_0(x_i)c_0+\sum_{i=1}^{n}\varphi_1(x_i)\varphi_1(x_i)c_1+\cdots+\sum_{i=1}^{n}\varphi_1(x_i)\varphi_m(x_i)c_m=\sum_{i=1}^{n}y_i\varphi_1(x_i)\\ \cdots\\ \sum_{i=1}^{n}\varphi_m(x_i)\varphi_0(x_i)c_0+\sum_{i=1}^{n}\varphi_m(x_i)\varphi_1(x_i)c_1+\cdots+\sum_{i=1}^{n}\varphi_m(x_i)\varphi_m(x_i)c_m=\sum_{i=1}^{n}y_i\varphi_m(x_i)\end{cases}$$

Итог: найдены $c_0,c_1,\dots,c_m\to\sum_{i=1}^{n}\rho(x_i)[y_i-\Phi_m(x_i)]^2$

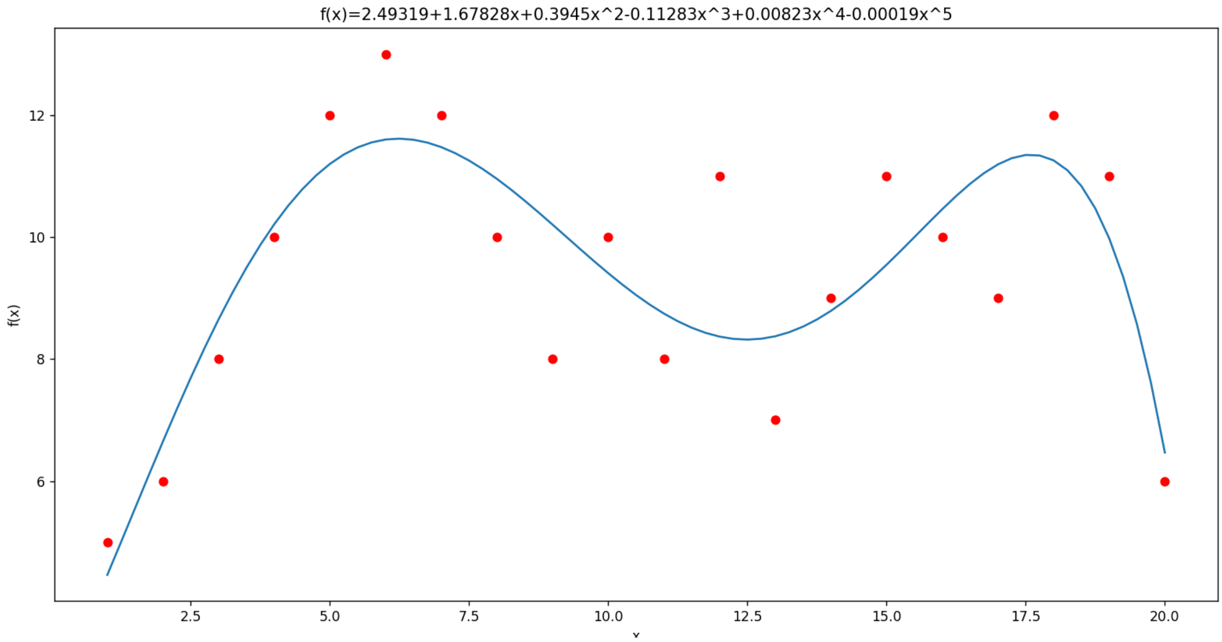
## Решение задачи

```
from numpy import linalg
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
y = [5, 6, 8, 10, 12, 13, 12, 10, 8, 10, 8, 11, 7, 9, 11, 10, 9, 12, 11, 6]

# Функция для вычисления коэффициентов аппроксимирующей кривой с полиномы по
m порядок
def coefficients(m):
    C = []
    b = []
    for i in range(m + 1):
        Ci = []
        bi = 0
        for j in range(m + 1):
            cij = 0
            for xi in x:
                cij = cij + xi ** (i + j)
            Ci.append(cij)
        for k in range(len(x)):
            bi = bi + (x[k] ** i) * y[k]
        C.append(Ci)
        b.append(bi)
    a = linalg.solve(C, b)
    return a

# Функция для вычисления значении многочлены с коэффициетами a в точке x
def f(x, a):
    resul = 0
    for k in range(len(a)):
        resul += a[k] * x ** k
    return resul

# Вычисление отклонений
a = coefficients(5)
s = 0
for i in range(len(x)):
    fi = f(x[i], a)
    s += (fi-y[i])**2

func = ""
for i in range(len(a)):
    ar = round(a[i], 5)
    if ar > 0:
        if i == 0:
            func += str(ar)
        elif i == 1:
            func += "+"+str(ar)+"x"
        else:
            func += "+"+str(ar)+"x^"+str(i)
    elif ar < 0:
        if i == 0:
            func += str(ar)
        elif i == 1:
            func += str(ar)+"x"
        else:
            func += str(ar)+"x^"+str(i)
print("Аппроксимирующая кривая: f(x) =", func)
print("Сумма квадратов отклонений: ", s)

lx = []
lf = []
```

```
for i in range(77):
    lx.append(1+i*0.25)
    lf.append(f(1+i*0.25, a))

# plt.plot(x, y)
plt.plot(lx, lf)
plt.title("f(x)="+func)
plt.xlabel("x")
plt.ylabel("f(x)")
for i in range(len(x)):
    plt.plot(x[i], y[i], 'ro')
plt.show()
```

## Пример выполнения программы

```
Аппроксимирующая кривая: f(x) = 2.49319+1.67828x+0.3945x^2-0.11283x^3+0.00823x^4-0.00019x^5
Сумма квадратов отклонений:  28.52497607655594
```




Аппроксимирующая кривая изображена на графике синем цветом.

## Заключение

В лабораторной работе требовалось решить задачу аппроксимации, а также построить график аппроксимирующей кривой и, дополнительно, подсчитать сумму квадратов отклонений между значениями функции и аппроксимирующей кривой в узловых точках. Для решения задачи была написана программа на языке программирования Python 3.9. По результатам расчетов построена аппроксимирующая кривая, изображенная на графике синем цветом.