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:
По курсу «Численные методы»

Работу выполнил: студент группы С18-712: Кольца И. В.
Проверил: Саманчук В.Н.

Москва 2020

Постановка задачи

Аппроксимировать таблично заданную функцию по методу наименьших квадратов, используя полиномы Чебышева по седьмой порядок включительно.

X	1	2	3	4	5	6	7	8	9	10
Y	5	6	8	10	12	13	12	10	8	10

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

Методика решения

Для решения поставленной задачи была написана программа на языке Python, в которой реализован полином Чебышева для аппроксимации таблично заданной функции методом наименьших квадратов.

Теоретическая справка

Метод наименьших квадратов используется:

- для решения переопределенных систем уравнений, когда количество уравнений превышает количество неизвестных;

- для поиска решения в случае обычных (не переопределенных) нелинейных систем уравнений;

- для аппроксимации точечных значений некоторой аппроксимирующей функцией.

Аппроксимирующая функция по методу наименьших квадратов определяется из условия минимума суммы квадратов отклонений (ξ_i) расчетной аппроксимирующей функции от заданного массива экспериментальных данных. Данный критерий метода наименьших квадратов записывается в виде следующего выражения:

$$\sum_{i=1}^{N} \xi_i^2 = \sum_{i=1}^{N} (F(x_i) - y_i)^2 \to \min$$

$F(x_i)$ - значения расчетной аппроксимирующей функции в узловых точках x_i,

y_i - заданный массив экспериментальных данных в узловых точках x_i.

Квадратичный критерий обладает рядом "хороших" свойств, таких, как дифференцируемость, обеспечение единственного решения задачи аппроксимации при полиномиальных аппроксимирующих функциях.

В зависимости от условий задачи аппроксимирующая функция представляет собой многочлен степени m

$$F_m(x) = a_0 + a_1 \cdot x + ... + a_{m-1} \cdot x^{m-1} + a_m \cdot x^m$$

Степень аппроксимирующей функции (m) не зависит от числа узловых точек, но ее размерность должна быть всегда меньше размерности (количества точек) заданного массива экспериментальных данных.

$$1 \le m \le N-1$$

· В случае если степень аппроксимирующей функции m=1, то мы аппроксимируем табличную функцию прямой линией (линейная регрессия).

· В случае если степень аппроксимирующей функции m=2, то мы аппроксимируем табличную функцию квадратичной параболой (квадратичная аппроксимация).

· В случае если степень аппроксимирующей функции m=3, то мы аппроксимируем табличную функцию кубической параболой (кубическая аппроксимация).

В общем случае, когда требуется построить аппроксимирующий многочлен степени m для заданных табличных значений, условие минимума суммы квадратов отклонений по всем узловым точкам переписывается в следующем виде:

$$S = \sum_{i=1}^{N} \xi_i^2 = \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + ... + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i\right)^2 \to \min$$

x_i, y_i - координаты узловых точек таблицы;

$a_j, (j = 0,...,m)$ - неизвестные коэффициенты аппроксимирующего многочлена степени m;

N - количество заданных табличных значений.

Необходимым условием существования минимума функции является равенству нулю ее частных производных по неизвестным переменным $a_j, (j = 0,...,m)$. В результате получим следующую систему уравнений:

$$\begin{cases} \frac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + ... + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i\right) = 0 \\ \frac{\partial S}{\partial a_1} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + ... + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i\right) \cdot x_i = 0 \\ ... \\ \frac{\partial S}{\partial a_m} = 2 \cdot \sum_{i=1}^{N} \left(a_0 + a_1 \cdot x_i + ... + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i\right) \cdot x_i^m = 0 \end{cases}$$

Преобразуем полученную линейную систему уравнений: раскроем скобки и перенесем свободные слагаемые в правую часть выражения. В результате полученная система линейных алгебраических выражений будет записываться в следующем виде:

$$\begin{cases} a_0 \cdot N + a_1 \cdot \sum_{i=1}^{N} x_i + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{m-1} + a_m \cdot \sum_{i=1}^{N} x_i^m = \sum_{i=1}^{N} y_i \\ a_0 \cdot \sum_{i=1}^{N} x_i + a_1 \cdot \sum_{i=1}^{N} x_i^2 + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^m + a_m \cdot \sum_{i=1}^{N} x_i^{m+1} = \sum_{i=1}^{N} y_i \cdot x_i \\ \ldots \\ a_0 \cdot \sum_{i=1}^{N} x_i^m + a_1 \cdot \sum_{i=1}^{N} x_i^{m+1} + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{2 \cdot m-1} + a_m \cdot \sum_{i=1}^{N} x_i^{2 \cdot m} = \sum_{i=1}^{N} y_i \cdot x_i^m \end{cases}$$

Данная система линейных алгебраических выражений может быть переписана в матричном виде:

$$\begin{vmatrix} N & \sum_{i=1}^{N} x_i & \cdots & \sum_{i=1}^{N} x_i^m \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \cdots & \sum_{i=1}^{N} x_i^{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^m & \sum_{i=1}^{N} x_i^{m+1} & \cdots & \sum_{i=1}^{N} x_i^{2 \cdot m} \end{vmatrix} \cdot \begin{vmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} y_i \cdot x_i \\ \vdots \\ \sum_{i=1}^{N} y_i \cdot x_i^m \end{vmatrix}$$

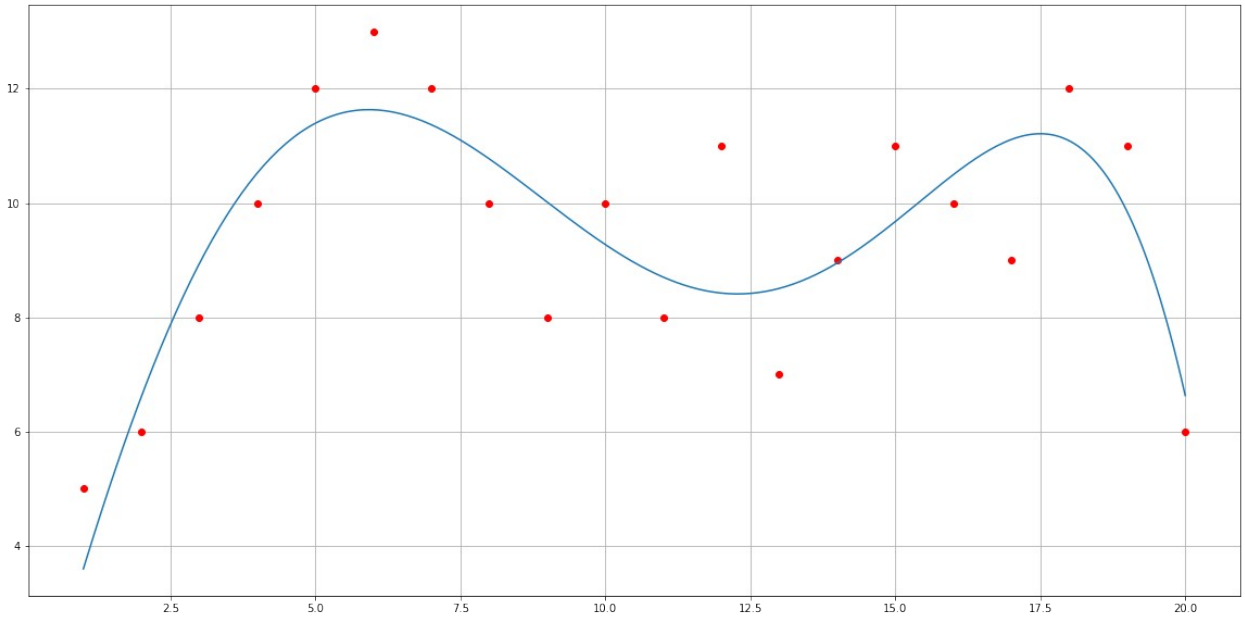
В результате была получена система линейных уравнений размерностью m+1, которая состоит из m+1 неизвестных. Данная система может быть решена с помощью любого метода решения линейных алгебраических уравнений (например, методом Гаусса). В результате решения будут найдены неизвестные параметры аппроксимирующей функции, обеспечивающие минимальную сумму квадратов отклонений аппроксимирующей функции от исходных данных, т.е. наилучшее возможное квадратичное приближение. Следует помнить, что при изменении даже одного значения исходных данных все коэффициенты изменят свои значения, так как они полностью определяются исходными данными.

Решение задачи

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

m = 7
X = np.arange(1, 21)
Y = np.array([5, 6, 8, 10, 12, 13, 12, 10, 8, 10, 8, 11, 7, 9, 11, 10, 9, 12, 11, 6])

def T(x, deg):
    if (deg < 0):
        return None

    if (deg == 0):
        return 1
    elif (deg == 1):
        return x
    else:
        return 2 * x * T(x, deg - 1) - T(x, deg - 2)

def chebval(x, coef):
    res = 0
    for i in range(len(coef)):
        res += coef[i] * T(x, i)
    return res

A = np.zeros((m + 1, m + 1))
B = np.array([0] * (m + 1))

for i in range(m + 1):
    for j in range(m + 1):
        A[i][j] = np.sum(T(X, i) * T(X, j))
    B[i] = np.sum(Y * T(X, i))

coef = np.linalg.solve(A, B)
x = np.linspace(1, 20, 300)
f = chebval(x, coef)

plt.figure(figsize=(20, 10))
plt.plot(X, Y, 'ro', x, f)
plt.grid(True)
print(f'Сумма квадратов отклонений = {np.sum((np.array(Y) - chebval(X, coef))**2)}')
```

Результат работы

```
Сумма квадратов отклонений = 29.742460028290953
```



Заключение

В работе требовалось аппроксимировать таблично заданную функцию по методу наименьших квадратов с помощью полинома Чебышева. Для решения задачи была написана программа на языке программирования Python.