Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Теоретическая информатика и компьютерные технологии»

Отчёт о лабораторной работе № 5

по курсу «Численные методы»

МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

Студент: К. Лозовска

Группа: ИУ9И-64Б

Преподаватель: А.Б.Домрачева

Москва, 2023

СОДЕРЖАНИЕ

ЦЕЛЬ И ПОСТАНОВКА ЗАДАЧИ

Цель: Аппроксимировать заданную функцию алгебраическими многочленами, применив метод наименьших квадратов.

Постановка задачи:

Аппроксимировать функцию по МНК многочленом третьей степени ($m = 4$). Найти: матрицу А и столбец b; набор коэффициентов $\lambda_1, \lambda_2, \lambda_3, \lambda_4$; значения аппроксимирующего многочлена $z(x)$ в средних точках отрезков между узловыми точками.

Тестовый пример:

1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
0.95	2.07	1.96	2.62	3.75	4.12	3.98	3.63	4.70

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Пусть значения приближаемой функции $y = f(x)$ заданы лишь в узлах $(x_i, y_i), i = 0, \dots, n$. часто значения y_i бывают известны со случайными ошибками. Тогда нет смысла проводить приближающую функцию точно через узлы (x_i, y_i), как делают при интерполяции. Вместо этого возникает задача аппроксимации: найти гладкую аналитически заданную функцию $z(x)$, доставляющую наименьшее значение величине

$$\text{СКУ} = \sqrt{\sum_{k=0}^{n} (z(x_k) - y_k)^2} = \sqrt{\sum_{k=0}^{n} \varepsilon_k^2}.$$

Эту величину называют среднеквадратичным уклонением (СКУ) функции $z(x)$ от системы узлов $(x_i, y_i), i = 0, \dots, n$, а описанный подход к решению задачи приближения функции – методом наименьших квадратов (МНК) (рис. 1).

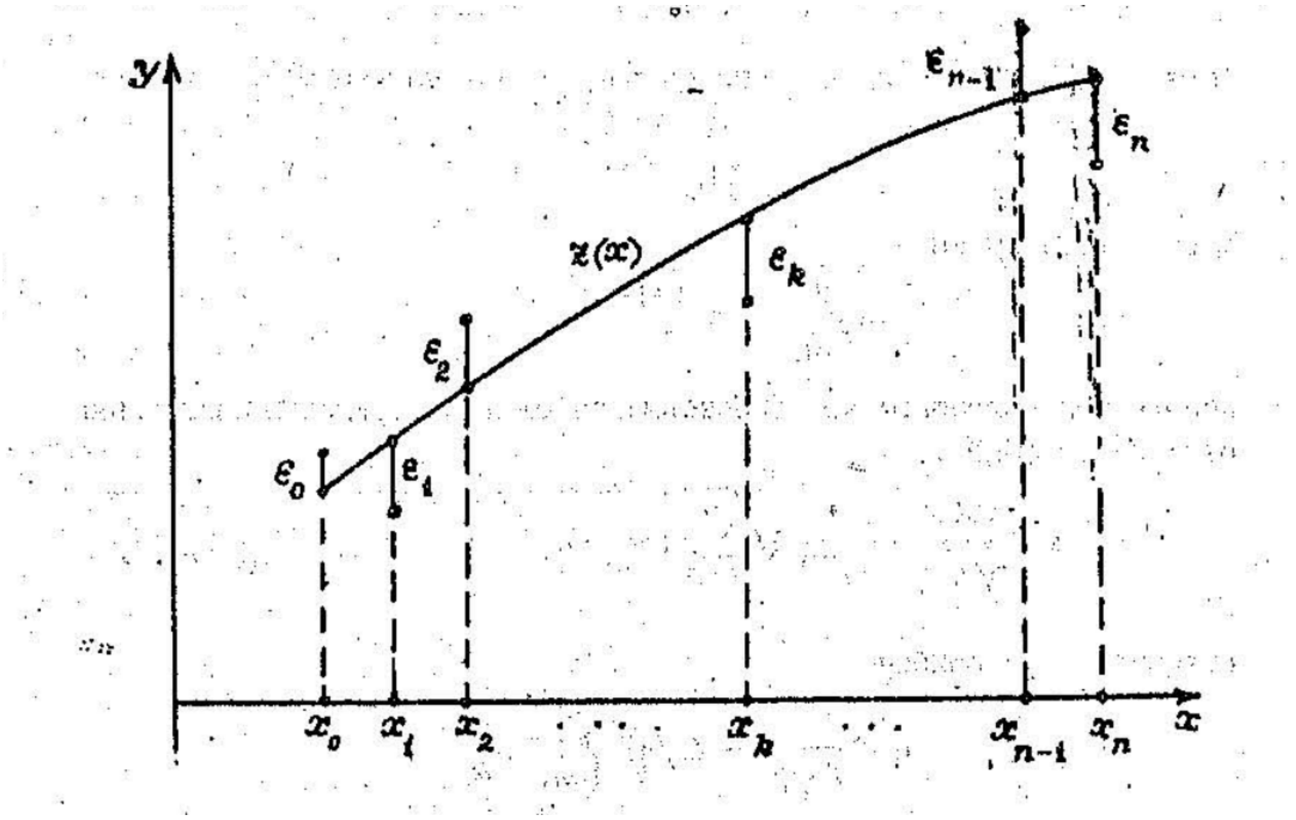

Рисунок 1

Как правило, $z(x)$ отыскивают в виде линейной комбинации заранее заданных функций:

$$z(x) = \lambda_1\varphi_1(x) + \cdots + \lambda_m\varphi_m(x).$$

Параметры $\lambda_i, i = 1, \dots, m$ являются решениями линейной системы наименьших квадратов

$$A\lambda = b,$$

где λ – столбец параметров λ_i, $A = (a_{ij})$ – симметричная положительно определенная матрица (матрица Грама) с коэффициентами $a_{ij} = \sum_{k=0}^{n} \varphi_i(x_i)\varphi_j(x_k)$, b – столбец правой части системы, $b_i = \sum_{k=0}^{n} \varphi_i(x_k)y_k$, $i, j = 1, \dots, m$.

Таким образом, система МНК имеет единственное решение $\lambda_1^*, \dots, \lambda_m^*$, дающее среднеквадратичному уклонению наименьшее значение (для всех функций данного вида). Решать систему рекомендуется методом квадратного корня.

Если приближаемая функция достаточно гладкая, хотя вид ее и неизвестен, аппроксимирующую функцию нередко ищут в виде алгебраического многочлена

$$z(x) = \lambda_1 + \lambda_2 x + \dots + \lambda_m x^{m-1}.$$

Тогда $\varphi_i = x^{i-1}$ и элементы матрицы Грама получают по формулам:

$$a_{ij} = \sum\nolimits_{k=0}^{n} x_k^{i+j-1},$$

а свободные члены –

$$b_i = \sum\nolimits_{k=0}^{n} y_k x_k^{i+j-1}, \qquad i, j = 1, \dots, m.$$

Абсолютная погрешностью аппроксимации служит среднеквадратичное отклонение (СКО):

$$\Delta = \frac{\text{СКУ}}{\sqrt{n}} = \frac{1}{\sqrt{n}} \sqrt{\sum_{k=0}^{n} (y_k - \lambda_1 - \lambda_2 x_k - \dots - \lambda_m x_k^{m-1})^2},$$

относительная ошибка: $\delta = \frac{\Delta}{||y||} = \Delta / \sqrt{\sum_{k=0}^{n} y_k^2}.$

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Данный код реализует метод наименьших квадратов для аппроксимации функции полиномом заданной степени.

Сначала задаются начальные данные: границы отрезка, количество точек, степень полинома, значения функции в узлах.

```
import numpy as np
from numpy import linalg as la

A = 1
B = 5
n = 9
m = 4
h = (B - A) / (n - 1)
y = [0.96, 2.07, 1.96, 2.62, 3.75, 4.12, 3.98, 3.63, 4.70]

x = []
x2 = []
a = []
b = [0] * m
```

Затем вычисляются коэффициенты полинома методом наименьших квадратов. Для этого создаются матрица системы уравнений и вектор свободных членов, которые заполняются в циклах. Затем находится обратная матрица и решение системы.

```
for i in range(n + 1):
    x.append(A + i * h)
    x2.append(A + (i + 1 / 2) * h)

for i in range(m):
    a.append([])
    for j in range(m):
        a[i].append(0)
        for k in range(n):
            a[i][j] += x[k] ** (i + j)
```

```
for i in range(m):
    b[i] = 0
    for k in range(n):
        b[i] += y[k] * (x[k] ** i)

for i in range(m):
    b[i] = [b[i]]
```

```
print("a: ", a)
print("b: ", b)

M1 = la.inv(a)
lambda_ = np.dot(M1, b)
print("lambda: ", lambda_)
```

Далее вычисляются значения полинома в узлах и считаются абсолютная и относительная погрешности.

```
z = []
z2 = []
for k in range(n):
    z_k = 0
    z2_k = 0
    for i in range(m):
        z_k += lambda_[i][0] * x[k] ** i
        z2_k += lambda_[i][0] * x2[k] ** i
    z.append(z_k)
    z2.append(z2_k)
```

```python
Delta = 0
for k in range(n):
    print(y[k], z[k], abs(y[k] - z[k]))
    Delta += (y[k] - z[k]) ** 2

Delta = (Delta ** (1 / 2)) / (n ** (1 / 2))
print("absolute error:", Delta)
```

```python
delta = 0
for k in range(n):
    delta += y[k] ** 2
delta = Delta / delta
print("relative error:", delta)
```

В конце выводятся результаты в виде таблицы.

```python
print("xt yt rest delta")
for k in range(n):
    print(f"x: {x[k]:.8f} y: {y[k]:.8f} result: {z[k]:.8f} delta: {abs(y[k] - z[k]):.8f}")
    if k != n - 1:
        print(f"x: {x2[k]:.8f} result: {z2[k]:.8f}")
```

РЕЗУЛЬТАТЫ

Для примера была использована следующая функция, заданная таблично:

1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
0.95	2.07	1.96	2.62	3.75	4.12	3.98	3.63	4.70

Результаты работы реализованного метода наименьших квадратов представлены в таблице 1.

Таблица 1. Результаты работы программы.

x	***y***	***y****	***delta***
1.00	*0.96*	*0.99424242*	*0.03424242*
1.25	-	*1.38274892*	-
1.50	*2.07*	*1.74287879*	*0.32712121*
1.75	-	*2.07569264*	-
2.00	*1.96*	*2.38225108*	*0.42225108*
2.25	-	*2.66361472*	

2.50	*2.62*	*2.92084416*	*0.30084416*
2.75	*-*	*3.15500000*	*-*
3.00	*3.75*	*3.36714286*	*0.38285714*
3.25	*-*	*3.55833333*	*-*
3.50	*4.12*	*3.72963203*	*0.39036797*
3.75	*-*	*3.88209957*	*-*
4.00	*3.98*	*4.01679654*	*0.03679654*
4.25	*-*	*4.13478355*	*-*
4.50	*3.63*	*4.23712121*	*0.60712121*
4.75	*-*	*4.32487013*	*-*
5.00	*4.70*	*4.39909091*	*0.30090909*

Вектор $\boldsymbol{\lambda}$: *[-0.8647619, 2.11712843, -0.26943723, 0.01131313]*

Абсолютная погрешность: *0.04255347*.

Относительная погрешность: *0.00362389*.

ВЫВОДЫ

В приведенном коде используется полином степени 4 для соответствия 9 точкам данных, чего может быть достаточно в зависимости от конкретной задачи. Также рассчитываются и отображаются абсолютные и относительные ошибки, что может указывать на точность результирующего полинома.

Для заданных значений x и y реализация метода, дает полином, который относительно хорошо соответствует данным. Расчетная абсолютная ошибка составляет 0,142, что означает, что в среднем разница между фактическими значениями y и прогнозируемыми значениями составляет около 0,142. Расчетная относительная ошибка составляет 0,0036, что означает, что ошибка составляет около 0,36% от общего значения y.

В целом точность метода наименьших квадратов зависит от качества и количества используемых данных.