Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки: 09.03.04 — Системное и прикладное программное обеспечение
Дисциплина «Вычислительная математика»

Лабораторная работа №4

Вариант 7

Выполнил:

Капарулин Тимофей Иванович

Преподаватель:

Машина Екатерина Алексеевна

г.Санкт-Петербург 2025 г.

Цель работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Вычислительная реализация

1) Таблица табулирования заданной функции

x	**y**
-2.0	-2.000
-1.8	-2.366
-1.6	-2.715
-1.4	-2.970
-1.2	-3.042
-1.0	-2.875
-0.8	-2.483
-0.6	-1.936
-0.4	-1.309
-0.2	-0.657
0.0	0.000

2) линейное приближение

$$\begin{cases} a\sum x^2_i + b\sum x_i = \sum x_i y_i \\ a\sum x_i + bn = \sum y_i \end{cases}$$

$$\sum x_i = -11$$
$$\sum x^2_i = 15.4$$
$$\sum y_i = -22.353$$
$$\sum x_i y_i = 27.089$$

Линейная модель: a = 1.076, b = -0.956

y = 1.076x - 0.956

3) квадратичное приближение

$$\begin{cases} a\sum x^4_i + b\sum x^3_i + c\sum x^2_i = \sum x^2_i y_i \\ a\sum x^3_i + b\sum x^2_i + c\sum x_i = \sum x_i y_i \\ \mathrm{a}\sum x^2_i + b\sum x_i + cn = \sum y_i \end{cases}$$

$$\sum x^3{}_i = -24.2$$
$$\sum x^4{}_i = 40.533$$
$$\sum x_i^2 y_i = -38.214$$

Квадратичная модель: a = 1.858, b = 4.794, c=0.16

$$y = 1.858x^2 + 4.794x + 0.16$$

4) среднеквадратическое отклонение

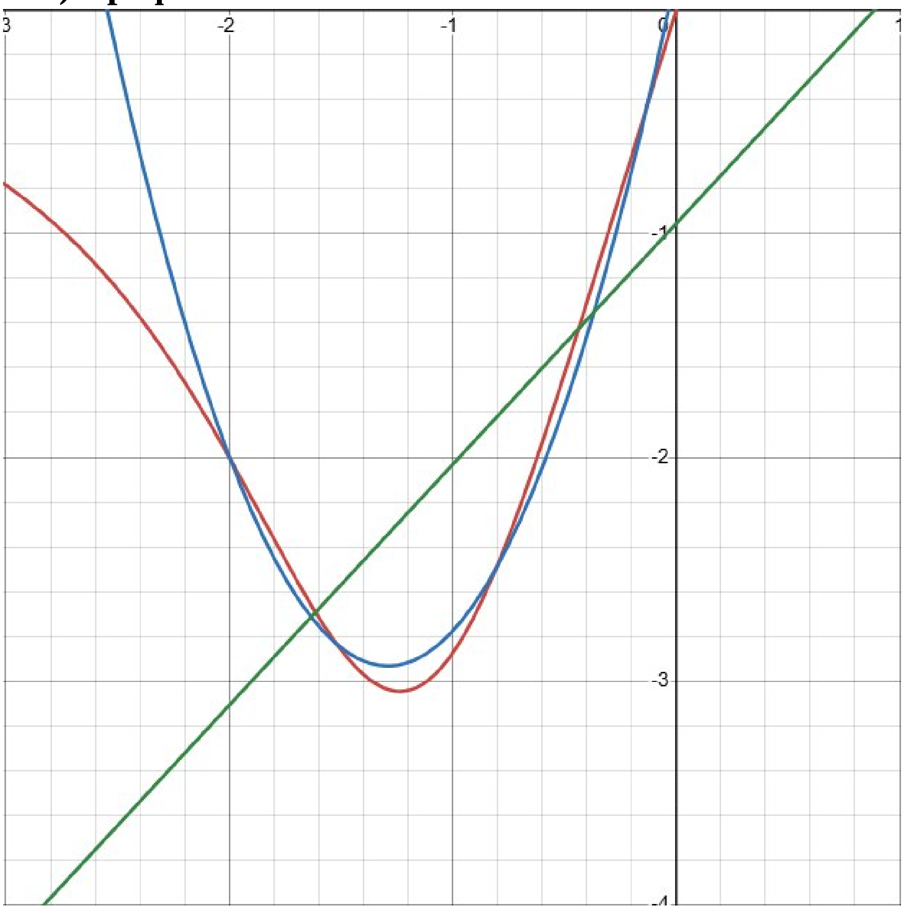
$$\sigma_{\text{лин}} = 0.663$$

$$\sigma_{\text{квад}} = 0.097$$

5) выбор наилучшего приближения

Квадратичная модель имеет меньшее СКО, поэтому она лучше приближает исходные данные.

6) графики



Синий – квадратичное приближение
Зеленый - линейное
Красный – исходная функция

Листинг програмы

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

X = []
Y = []

# Функции для аппроксимации
def linear(x, a, b):
    return a * x + b

def quadratic(x, a, b, c):
    return a * x**2 + b * x + c

def cubic(x, a, b, c, d):
    return a * x**3 + b * x**2 + c * x + d

def exponential(x, a, b):
    return a * np.exp(b * x)

def logarithmic(x, a, b):
    return a * np.log(x + 1e-9) + b  # +1e-9 для избежания log(0)

def power(x, a, b):
    return a * (x + 1e-9)**b  # +1e-9 для избежания 0^b

# Метод наименьших квадратов для всех моделей
models = {
    "Линейная": (linear, 2),
    "Квадратичная": (quadratic, 3),
    "Кубическая": (cubic, 4),
    "Экспоненциальная": (exponential, 2),
    "Логарифмическая": (logarithmic, 2),
    "Степенная": (power, 2),
}
class Main:
    def __init__(self):
        point_nmb = 0

        while(point_nmb not in [8, 9, 10, 11, 12]):
            print("Введите количество вводимых точек(8-12): ", end="")
            point_nmb = self.__get_param("equation number", int, input)
            print()
```

```python
print("Введите путь до данных(\"\" - ввод в терминале): ", end="")
file = self.__get_param("file name", str, input)

if file:
    with open(file) as f:
        for i in f.readlines():
            xi, yi = map(float, i.split(" "))
            X.append(xi)
            Y.append(yi)
else:
    for i in range(point_nmb):
        xi, yi = map(float, input().split(" "))
        X.append(xi)
        Y.append(yi)


if len(X) != point_nmb:
    print("Error: invalid point's number")
    exit()

x = np.array(X)
y = np.array(Y)

# решение
results = {}
for name, (func, params) in models.items():
    try:
        popt, pcov = curve_fit(func, x, y, maxfev=10000)
        y_pred = func(x, *popt)
        S = np.sum((y_pred - y)**2)
        n = len(x)
        mse = np.sqrt(S / n)
        results[name] = {
            "coeffs": popt,
            "S": S,
            "mse": mse,
            "y_pred": y_pred,
        }
    except Exception as e:
        print(f"Ошибка в модели {name}: {str(e)}")

# Коэффициент корреляции Пирсона (для линейной)
if "Линейная" in results:
    r = np.corrcoef(x, y)[0, 1]
    results["Линейная"]["r"] = r
    R2 = r**2
    results["Линейная"]["R2"] = R2
```

```python
    # Вывод результатов
    print("Результаты аппроксимации:")
    for name, data in results.items():
        print(f"\n--- {name} ---")
        print(f"Коэффициенты: {np.round(data['coeffs'], 4)}")
        print(f"S = {data['S']:.4f}")
        print(f"СКО = {data['mse']:.4f}")
        if name == "Линейная":
            print(f"Коэф. корреляции Пирсона: {data['r']:.4f}")
            print(f"Коэф. детерминации R²: {data['R2']:.4f}")

    # Выбор наилучшей модели
    best_model = min(results.items(), key=lambda x: x[1]["mse"])
    print(f"\nНаилучшая модель: {best_model[0]} (СКО = {best_model[1]['mse']:.4f})")

    # Построение графиков
    plt.figure(figsize=(12, 8))
    plt.scatter(x, y, label="Исходные данные", color="black")

    x_plot = np.linspace(x.min() - 0.5, x.max() + 0.5, 100)
    for name, data in results.items():
        if name == "Линейная":
            y_plot = linear(x_plot, *data["coeffs"])
        elif name == "Квадратичная":
            y_plot = quadratic(x_plot, *data["coeffs"])
        elif name == "Кубическая":
            y_plot = cubic(x_plot, *data["coeffs"])
        elif name == "Экспоненциальная":
            y_plot = exponential(x_plot, *data["coeffs"])
        elif name == "Логарифмическая":
            y_plot = logarithmic(x_plot, *data["coeffs"])
        elif name == "Степенная":
            y_plot = power(x_plot, *data["coeffs"])
        plt.plot(x_plot, y_plot, label=name)

    plt.xlabel("x")
    plt.ylabel("y")
    plt.title("Аппроксимация функции")
    plt.legend()
    plt.grid(True)
    plt.savefig("graph.png")

# получение 1 параметра
def __get_param(self, name, type, func):
    try:
        return type(func())
    except:
```

```
        print(f"Incorrect {name}")
        exit()

if __name__ == "__main__":
    Main()
```

Выводы

В данной работе были реализованы методы аппроксимации функции. Методы были протестированы на различных примерах. Результаты показали, что реализованные алгоритмы успешно справляется с поставленной задачей и находят решения в пределах допустимых погрешностей.